PR1ME

engineering handbook engineering

handbook engineering handbook

book engineering handbook engin

engineering handbook engineering

book engineering handbook engin

engineering handbook engineering

ok engineering handbook engine

engineering handbook engineering

book **engineering handbook** engin

handbook engineering handbook

engineering handbook engineering

PRIME ENGINEERING HANDBOOK

for

PRIME P400, P500; PRIMOS IV, V

June, 1978

PRIME ENGINEERING HANDBOOK

This revision corresponds to PRIMOS REV 15.

This is the initial release of the Prime Engineering Handbook, a document produced and maintained by Prime Computer Research and Development. Comments and requests to be added to the distribution list for handbook updates should be addressed to Maggie Chianese, Documentation Administrator, Prime Computer Research and Development, 3 Newton Executive Park, Newton, MA 02162 (617-964-1730, x215).

Published by Prime Computer, Incorporated
145 Pennsylvania Ave., Framingham, MA 01701

June 21, 1978

All Rights Reserved.

The information contained in this handbook is subject to change without notice. Prime Computer Incorporated assumes no responsibility for errors that may appear in this document. This handbook is intended for the use of Prime employees only.

Copyright 1978 by Prime Computer, Inc.

Table of Contents

1 INTRODUCTION

OVERVIEW

This handbook provides a summary of information needed for the development and maintenance of Prime P4ØØ and P5ØØ hardware and software systems. While this book contains information useful to a general user community, the information is presented in very condensed form. It is assumed that the reader has had prior contact with this material and, therefore, that detailed descriptions are unnecessary.

Some of the information contained herein pertains only to the latest revision of PRIMOS. This information will be updated on a regular basis as new revisions are released. (Refer to the inside of the cover for the revision currently reflected in this version of the handbook.) Readers are strongly urged to report errors, discrepancies, and omissions as soon as they are noticed.

SYNTAX

The descriptions of commands and external programs use the following syntax:

Abbreviations

Uppercase letters represent minimal abbreviations for commands and options. (When actually typing the command or option, either uppercase or lowercase can usually be used. Exceptions are noted.) For example:

```
COMOutput
```

specifies the COMOUTPUT command. COMO, COMOU, COMOUTP, etc., also specify this command.

Command Line Variables

Greater-than (>) and lesser-than (<) signs surrounding a string indicate a variable for which specific information is to be substituted; for example:

```
<filename>
```

should be replaced with a valid filename. Notice, however, that the use of greater-than and lesser-than signs in treenames is literal.

Optional Parameters

Brackets enclose optional parameters for a command; for example:

```
SHutdn [ ALL ]
```

Alternative Operand Specification, Defaults

When an operand has more than one possible specification, choices are stacked. A default option, if any, is underscored; for example:

```
OPRpri [ 1 ]
       [ Ø ]
```

The OPRPRI command accepts a single parameter of 1 or Ø. If none is specified, the default parameter is Ø.

Repeated Operands

Ellipsis indicate an operand that may be repeated one or more times; for example:

```
Close <funit> ...
```

The CLOSE command accepts one or more file unit specifications (separated by blanks or commas).

2 CENTRAL PROCESSING UNIT

ARGUMENT POINTER (AP)

```
 1                  16 1                    16
|B|BBB|I-Z|ZLS|---|---|P|PPP|PPW|WWW|WWW|WWW|
```

BBBB Bit Number
I Indirect
ZZ Base Register: 00 - PB
01 - SB
10 - LB
11 - XB
L Last AP in arglist
S Store
P..W Page/Word Disp from Base Reg

Typical values, first word:

100	PB Rel.	4100	PB Rel. Indirect
300	PB Rel. Last	4300	PB Rel. Indirect Last
500	SB Rel.	4500	SB Rel. Indirect
700	SB Rel. Last	4700	SB Rel. Indirect Last
1100	LB Rel.	5100	LB Rel. Indirect
1300	LB Rel. Last	5300	LB Rel. Indirect Last
1500	XB Rel.	5500	XB Rel. Indirect
1700	XB Rel. Last	5700	XB Rel. Indirect Last

CHECKS

CHECK HEADER (4 WORDS): PBH,PBL,KEYS,MODALS

4/200	Power Fail
4/270	Memory Parity
4/300	Machine Check
4/310	Missing Memory Module

On entry to fault handler, mode=64V, MCM=0 for all but ECCC, for which MCM=MCM-at-check - 1.

MMOD interrupts any other check in progress.

MCHK and ECCU interrupt ECCU in progress if MCM=2 (QUIET).

CONCEALED STACK/QUEUE

(Valid only between time of fault and subsequent CALF instruction.)

PCB+'74 --> FIRST	+0	PBH
PCB+'75 --> NEXT	+1	PBL
PCB+'76 --> LAST	+2	KEYS
	+3	FCODE
	+4	FADDRH
	+5	FADDRL

(PB, KEYS are those of procedure causing the fault.)

DIAGNOSTIC STATUS WORD (DSW)

DSWRMA: R34 DSWSTAT: R35 DSWPB: R36

DSWSTATH:

1	100000	CI - Check Immediate
2	040000	MC - Machine Check
3	020000	MP - Memory Parity (ECC)
4	010000	MM - Missing Memory
5-7	007000	Machine Check Code (Valid if bit 8=1):
		xx0xxx Peripheral Data (BPD) Output
		xx1xxx Peripheral Addr (BPA) Input
		xx2xxx Memory Data (BMD) Output
		xx3xxx Cache Data (RCD)
		xx4xxx Peripheral Addr (BPA) Output
		xx5xxx RDX-BPD Input
		xx6xxx Memory Address (BMA)
		xx7xxx Register File (RF)
8	000400	Not RCM Parity (P500, XCS)
9	000200	ECCU -- ECC Uncorrectable Error
10	000100	ECCC -- ECC Correctable Error
11	000040	BUP Invalid -- RP Backup Count Invalid
12-14	000034	RP Backup Count -- Sub from DSWPB
15	000002	Check During DMX
16	000001	IO Bus -- DMX, PIO, u-code check

DSWSTATL:

1-5	174000	ECCC Syndrome*:

000xxx	MB	100xxx	MB
004xxx	MB	104xxx	7
010xxx	MB	110xxx	MB
014xxx	15	114xxx	3
020xxx	MB	120xxx	MB
024xxx	14	124xxx	2
030xxx	13	130xxx	1
034xxx	9	134xxx	C2
040xxx	MB	140xxx	8
044xxx	MB	144xxx	6
050xxx	MB	150xxx	5
054xxx	12	154xxx	C5
060xxx	16	160xxx	4
064xxx	11	164xxx	C4
070xxx	10	170xxx	C3
074xxx	RP,Cl	174xxx	NE

6	002000	OP -- Overall Parity
7	001000	Unused
8	000400	Mod # (L.O. addr bit of module in err)
9	000200	RMA Invalid
10	000100	Unused
11-16	000077	U-Verify Test Number

*** U-Verify Test Number ***
to be supplied at next update

* MB=Multibit, RP,LP=Right/Left Parity,
Cn=Check Bit n, NE=No Errors

See Micro Code Handbook for additional information.

DMQ

See DMX under Peripheral I/O.

DESCRIPTOR TABLE ADDRESS REGISTER (DTAR)

```
 1                   16 1                  16
|S|SSS|SSS|SSS|AAA|AAA|A*|A*AA|AAA|AAA|AAA|AAA|
```

S...S   1024 - # SDWs in Table.
A...A   High Order 21 bits of Physical Addr of Table.
        (Bit 22 taken as zero.)

*: bits 1 and 2 of the second word must be equal.

ENTRY CONTROL BLOCK (ECB)

```
    ______________________________
 0 |          PROCEDURE           |
 1 |            BASE              |
 2 |       STACK FRAME SIZE       |
 3 |       SN OF STACK ROOT       |   (0 => USE CURRENT)
 4 |  DISP OF ARGLIST IN S.F.     |
 5 |    NUMBER OF ARGUMENTS       |
 6 |            LINK              |
 7 |            BASE              |
10 |            KEYS              |
```

Locations '11 through '17 are set to 0.

FAULTS

FCODEH: CRS 26H   FADDR: CRS 27

FAULT	#	OFFSET	VECT	FCODEH	FADDR	RING	SAVED PB
RXM	0	0	62	---	addr	curr	backed
PROCESS	1	4	63	ABFLAGS	---	0	curr
PAGE	2	10	64	---	addr	0	backed
SVC	3	14	65	---	---	curr	backed
UII	4	20	66	cur PBL	addr	curr	backed
ILL	10	40	72	cur PBL	addr	curr	backed
ACCESS	11	44	73	code	addr	0	backed
ARITH	12	50	74	code	addr	curr	curr
STACK	13	54	75	code	addr	0	backed
SEGMENT	14	60	76	code	addr	0	backed
POINTER	15	64	77	code	ptr addr	curr	backed

INSTRUCTION SET

See Section 5.

INDIRECT POINTER (IP)

```
 1                   16 1                  16
|F|RRE|SSS|SSS|SSS|SSS|P|PPP|PPW|WWW|WWW|WWW|
```

F   1 => Missing Pointer
RR   Ring Number (00-11)
E   1 => Word 3 = bit number: BBBB------------
S..S   Segment Number
P..W   Page Number/Word Number

(As effective address in a base reg, F,E ignored.)

KEYS, MODALS

(CRS 24 RFILE 124,164 CRASH 50,150)

KEYSH (Keys):

1	100000	C Bit
2	040000	Double Precision
3	020000	Link Bit (L)
4-6	016000	Addressing Mode:
		x00xxx 16S
		x02xxx 32S
		x04xxx 64R
		x06xxx 32R
		x10xxx 32I
		x14xxx 64V
7	001000	FLEX (0 => Allow Fault)
8	000400	IEX (Integer Exception)
9	000200	CCLT (condition code)
10	000100	CCEQ (condition code)
11-14	000074	Unused
15	000002	ID (In Dispatcher)
16	000001	SD (Save Done)

KEYSL (Modals):

1	100000	ENB (1 => Enable Interrupts)
2	040000	VIM (1 => Vectored Int. Mode)
3-8	037400	Unused
9-11	000340	CRS: xxx00x => Reg File 2
		xxx04x => Reg File 3
12	000020	MIO (1 => Mapped I/O)
13	000010	PXM (1 => Process Exchange Mode)
14	000004	SEG (1 => Segmentation Mode)
15-16	000003	MCM (Machine Check Mode):
		xxxxx0 None
		xxxxx1 Memory Parity
		xxxxx2 Quiet
		xxxxx3 Record

MODALS

See KEYS (KEYSL).

PAGE MAPS (HMAP, LMAP)

Starts at 4/4000. HMAP, LMAP interleaved in 64-word chunks, thus 128 words/segment in system.

HMAP (Hardware Map):

1	100000	1 => Page Resident.
2	040000	1 => Page Referenced.
3	020000	0 => Page Modified.
4	010000	1 => Shared Page (inhibits cache).
5-16	007777	H.O. 12 bits of physical page addr. (Bits 13-22 taken as zero.)

If non-resident, bits 3,5 software defined:

3,5	024000	Page status:
		000000 Not resident, copy on disk
		020000 Not resident, no copy on disk
		004000 In transition, coming in
		024000 In transition, going out

LMAP (Software Map -- HMAP+'100):

1-2	140000	Lock ctr 0 = unlock, not 0 = locked
3	020000	First Time (just paged in)
4	010000	Use alternate paging device
5-16	007777	Record index (1 val/8 pages)

SEGMENT	PAGE MAP LOCATION
0	4000
1	4200
4	4400
5	4600
6	5000
7	5200
10	5400
11	5600
12	6000
6000	6200

PANEL

See MEMORY/REGISTER DISPLAY under OPERATIONAL PROCEDURES.

PROCESS CONTROL BLOCK (PCB)

(See also PCBs under PRIMOS IV.)

Ø	LEVEL (IN READY LIST)
1	LINK (NEXT PCB IN LIST)
2	WAIT LIST SN (Ø=>READY)
3	WAIT LIST WORD NUMBER
4	ABORT FLAGS
5	RESERVED
6	RESERVED
7	RESERVED
1Ø	ELAPSED TIMER (LOW)
11	ELAPSED TIMER (HIGH)
12	DTAR2H
13	DTAR2L
14	DTAR3H
15	DTAR3L
16	INTERVAL TIMER
17	RESERVED
2Ø	SAVE MASK
21	KEYS
22	GRØ
23	"
24	GR1
25	"
26	GR2
27	"
3Ø	GR3
31	"
32	GR4
33	"
34	GR5
35	"
36	GR6
37	"
4Ø	GR7
41	"
42	FPØ
43	"
44	"
45	"
46	FP1
47	"
5Ø	"
51	"
52	PBH
53	PBL
54	SBH
55	SBL
56	LBH
57	LBL
6Ø	XBH
61	XBL
62	FAULT VECTOR, RING Ø
63	"
64	FAULT VECTOR, RING 1
65	"
66	RESERVED
67	"
7Ø	FAULT VECTOR, RING 3
71	"
72	PAGE FAULT VECTOR
73	"
74	CONCEALED STACK FIRST
75	CONCEALED STACK NEXT
76	CONCEALED STACK LAST
77	RESERVED

READY LIST

PPA current level/current PCB
PPB next level/next PCB
LEVELn --> FIRST PCB ON LEVELn
LEVELn+1 --> LAST PCB ON LEVELn
PCB+Ø --> LEVEL THIS PCB IS ON
PCB+1 --> NEXT PCB, Ø IF LAST

Ready list in Segment 4, starting at 4/6ØØ:

LEVEL	PCBS ON LEVEL
6ØØ	CLOCK PROCESS
6Ø2	SMLC PROCESS
6Ø4	AMLC PROCESS
6Ø6	MPC, MP2 PROCESSES
61Ø	VERSATEC PROCESS
612	IPC PROCESS
614	RINGNET PROCESS
616	SPARE1, SPARE2 PROCESSES
62Ø	SUPERVISOR PROCESS (USER 1)
622	PRIORITY 3 USER PROCESSES
624	PRIORITY 2 USER PROCESSES
626	PRIORITY 1 USER PROCESSES (NORMAL LEVEL)
63Ø	PRIORITY Ø USER PROCESSES
632	BACKSTOP PROCESS

RSAV FORMAT

Registers as saved/restored by the RSAV/RRST instructions.

```
     _____________
 Ø  | SAVE MASK  |
 1  |    FRN1    |
    |____________|
 3  |    FR1     |
    |____________|
 5  |    FRNØ    |
    |____________|
 7  |    FRØ     |
    |____________|
11  |    GR7     |
    |____________|
13  |    GR6     |
    |____________|
15  |    GR5     |
    |____________|
17  |    GR4     |
    |____________|
21  |    GR3     |
    |____________|
23  |    GR2     |
    |____________|
25  |    GR1     |
    |____________|
27  |    GRØ     |
    |____________|
31  |   X-BASE   |
    |____________|
```

SAVE MASK (1=>REG SAVED):

1-4	17ØØØØ	Unused
5	ØØ4ØØØ	FRN1
6	ØØ2ØØØ	FR1
7	ØØ1ØØØ	FRNØ
8	ØØØ4ØØ	FRØ
9	ØØØ2ØØ	GR7
1Ø	ØØØ1ØØ	GR6
11	ØØØØ4Ø	GR5
12	ØØØØ2Ø	GR4
13	ØØØØ1Ø	GR3
14	ØØØØØ4	GR2
15	ØØØØØ2	GR1
16	ØØØØØ1	GRØ

(XB always saved.)

SECTOR Ø (P3ØØ only)

Ø	X Register (Index Register)
1	A Register (Arithmetic, Shifts, I/O)
2	B Register (Ext Arithmetic, Shifts)
3	Stack Pointer
4	FLPH (Floating Point High)
5	FLPL (Floating Point Low)
6	VSC (Visible Shift Counter)
7	P Register (Program Counter)
1Ø	PMAR (Page Map Address Register)
11	Microcode Scratch Location
12	EAS (Effective Addr Save - ILL, UII, Ints)
13	Microcode Scratch Location
14	Y Register Save (Control Panel, DMA)
15	M Register Save (Control Panel, DMA)
16	Microcode Scratch Location
17	Microcode Scratch Location
2Ø-37	DMA Range/Start Address Pairs
4Ø-47	Reserved for DMC Channel Pairs
6Ø	PFI (Power Fail Int, Watchdog Timer)
61	RTCI (Real Time Clock Increment)
62	REVI (Restricted Execution Violation Int.)
63	Standard Interrupt (Compatible Mode)
64	Page Fault Interrupt
65	SVC Interrupt
66	UII (Unimplemented Instruction Interrupt)
67	Memory Data Parity Error
7Ø	Machine Check
71	Missing Memory Module
72	Illegal Instruction Interrupt
73	Page Write Violation
74	FLEX (Floating Point Exception)
75	Procedure Stack Underflow (3ØØ Only)
76-1ØØ	Debugging Scratch Area
1Ø1-177	Interrupt Vectors (Vectored Interrupt Mode)
2ØØ-777	General Cross Sector Links

SEGMENT DESCRIPTOR WORD (SDW)

```
 1                    16 1                   16
|P|PPP|PPP|PPP|ØØØ|ØØØ|F|AAA|BBB|CCC|PPP|PPP|
 7                    22               1     6
```

P...Ø	Page number/Word number of pagemap		
F	Fault, 1 => No segment or missing pagemap		
AAA	Access controls for Ring 1:	ØØØ	No access
BBB	Access controls for Ring 2	ØØ1	Gate
CCC	Access controls for Ring 3	Ø1Ø	Read
		Ø11	Read/Write
		1ØØ	Reserved
		1Ø1	Reserved
		11Ø	Read/XEQ
		111	R/W/XEQ

REGISTERS

RFIL ADDR = Address in Register File
CRASH ADDR = Disp in hardware register save area.

RFIL ADDR	HIGH	LOW	CRASH ADDR	RFIL ADDR	HIGH	LOW	CRASH ADDR
Ø	TRØ	-	300	40			200
1	TR1	-	302	41			202
2	TR2	-	304	42			204
3	TR3	-	306	43			206
4	TR4	-	31Ø	44			21Ø
5	TR5	-	312	45			212
6	TR6	-	314	46			214
7	TR7(PB)	-	316	47			216
1Ø	GRMX1	-	32Ø	5Ø			22Ø
11	GRMX2	-	322	51			222
12		RATMPL	324	52			224
13	RSGT1	-	326	53			226
14	RSGT2	-	33Ø	54			23Ø
15	RECC1	-	332	55			232
16	RECC2	-	334	56			234
17		REOIV	336	57			236
2Ø	ZERO	ONE	34Ø	6Ø	(2Ø)	(21)	24Ø
21	PBSAVE	-	342	61			242
22	GRMX3	-	344	62	(22)	(23)	244
23	GRMX4	-	346	63			246
24	C377		35Ø	64	(24)	(25)	25Ø
25			352	65			252
26			354	66	(26)	(27)	254
27			356	67			256
3Ø	PSWPB	-	36Ø	7Ø	(3Ø)	(31)	26Ø
31	PSWKEYS	-	362	71			262
32	PLA:PPA	PCBA	364	72	(32)	(33)	264
33	PLB:PPB	PCBB	366	73			266
34	DSWRMA	-	37Ø	74	(34)	(35)	27Ø
35	DSWSTAT	-	372	75			272
36	DSWPB	-	374	76	(36)	(37)	274
37	RSAVPTR	-	376	77			276

() indicate P3ØØ address mapping

TR7	PB at machine halt
PSWPB	PB at last interrupt
PSWKEYS	Keys at last interrupt
PPA	Current level/current PCB
PPB	Next level/next PCB
RSAVPTR	Reg save area ptr. Ø => regs saved.

See Micro Code Handbook for additional information.

CRS ADDR	HIGH	LOW	RF2 ADDR	CRASH ADDR2	RF3 ADDR	CRASH ADDR3
Ø	GRØ:OLTL	-	1ØØ	Ø	14Ø	1ØØ
1	GR1:PTS	-	1Ø1	2	141	1Ø2
2	GR2(1,A,LH)	-(2,B,LL)	1Ø2	4	142	1Ø4
3	GR3(EH)	-(EL)	1Ø3	6	143	1Ø6
4	GR4	-	1Ø4	1Ø	144	11Ø
5	GR5(3,S,Y)	-	1Ø5	12	145	112
6	GR6	-	1Ø6	14	146	114
7	GR7(Ø,X)	-	1Ø7	16	147	116
1Ø	FAR1(13)	-	11Ø	2Ø	15Ø	12Ø
11	FLR1	-	111	22	151	122
12	FAR2(4)	-(5)	112	24	152	124
13	FLR2(6)	-	113	26	153	126
14	PB	-(Ø=>CRS)*	114	3Ø	154	13Ø
15	SB(14)	-(15)	115	32	155	132
16	LB(16)	-(17)	116	34	156	134
17	XB	-	117	36	157	136
2Ø	DTAR3(1Ø)	-	12Ø	4Ø	16Ø	14Ø
21	DTAR2	-	121	42	161	142
22	DTAR1	-	122	44	162	144
23	DTARØ	-	123	46	163	146
24	KEYS	MODALS	124	5Ø	164	15Ø
25	OWNER	-	125	52	165	152
26	FCODE(11)	-	126	54	166	154
27	FADDR	-(12)	127	56	167	156
3Ø	TIMER	-	13Ø	6Ø	17Ø	16Ø
31			131	62	171	162
32			132	64	172	164
33			133	66	173	166
34			134	7Ø	174	17Ø
35			135	72	175	172
36			136	74	176	174
37			137	76	177	176

* Current PBL at halt in TR7L -- R7.

SEMAPHORES

+Ø # WAITS - # NOTIFIES, e.g., <Ø => notifies outstanding, >Ø => processes waiting, =Ø => wait list empty.

+1 Pointer to first waiting PCB on queue.

STACK FRAME, STACK ROOT

```
 Ø |_Ø=>PCL 1=>CALF|
 1 |_SN of STK ROOT|
 2 |    PB FOR     |
 3 |____RETURN_____|
 4 |  CALLER'S SB  |
 5 |_______________|
 6 |  CALLER'S LB  |
 7 |_______________|
10 |_CALLER'S KEYS_|
11 |______PBCL_____|
12 |_FCODE if CALF_|  <--- START OF AUTOMATIC
13 | FADDR if CALF |       STORAGE IF PCL (*)
14 |_______________|
```

*: First Argument Ptr pointed to by ECB+4.

STACK ROOT HEADER (WORD Ø OF SEGMENT)

Ø,1 - FREE POINTER (SN/WN)
2,3 - FIRST EXTENSION (SN/WN)

STACK EXTENSION HEADER (WORD Ø OF SEGMENT)

Ø,1 - Ø/Ø
2,3 - NEXT EXTENSION (SN/WN)

WAIT LIST

SEMAPHORE+Ø = COUNT OF WAITING PCBS
SEMAPHORE+1 --> FIRST PCB

PCB+1(LINK) --> NEXT PCB ON WAIT LIST
LAST PCB+1 --> Ø

PCB+2,3 --> SEMAPHORE PROCESS IS WAITING ON

Note: wait list ordered by increasing ready list level, i.e., highest priority processes first in list.

3 COMMANDS

ADDISK — ADD DISKS TO SYSTEM

```
ADdisk <dvno> ...
```

System user only.
Internal command.

AMLC -- SET AMLC LINE CHARACTERISTICS

```
AMlc [ Tty  ] <line> [<config>] [<lword>]
     [ TRan ]
     [TTYHs ]
     [TRANHS]
     [TTYNop]
```

If 2≤USER≤TRMUSR, <line>=USER-2.

<config>:
2Ø33 - 1ØØ BAUD
2213 - 3ØØ BAUD
2313 - 12ØØ BAUD
2413 - 96ØØ BAUD

<lword>:

bit		Meaning when on
1	1ØØØØØ	Half duplex
2	Ø4ØØØØ	No LF after CR
3	Ø2ØØØØ	XOFF/XON Recognition
4	Ø1ØØØØ	XOFF Received
5-8	ØØ74ØØ	Reserved
9-16	ØØØ377	User number

System user only.
Internal command.

ASRCWD -- SET VIRTUAL ASR CONTROL WORD

```
ASRcwd [ Ø ]  Terminal (Port 1)
       [ 2 ]  Centronics Printer #2 (Port 3)
       [ 4 ]  Centronics Printer #1 (Port 2)
```

Internal command.

ASSIGN -- ASSIGN PERIPHERAL DEVICE

```
ASsign AMLC [<protocol>] <line> [<config>] [<lword>]
       Cenpr  [-WAIT]
       CE2pr  [-WAIT]
       CArdr  [-WAIT]
       Ptr    [-WAIT]
       PUnch  [-WAIT]
       PRn    [-WAIT]     (where n=Ø,1)
       CRl    [-WAIT]
       MTn    [-WAIT]     (where n=Ø,7)
       SMLCnn [-WAIT]     (where nn=ØØ,Ø3)
       PLot   [-WAIT]
       Disk   <dvno> [-WAIT]
       SMLC   [-WAIT] <line>
```

<config> and <lword> described under AMLC on previous page.

<protocol>:
Tty
TTYHs
TRan
TRANHs
TTYNop

Internal command.

ATTACH -- ATTACH TO UFD

```
Attach [<ufdname>] [<passwd>] [<dvno>] [<option>]
                              [1ØØØØØ] [   2    ]
                              [177777] [   1    ]
                                       [ 177777 ]
```

<dvno>:
1ØØØØØ => search all started devices,
177777 => search MFD of current device

<option>:
2 => set home UFD after attach to subUFD
1 => dont set home UFD after attach to subUFD
177777 => attach to UFD and dont set home

Internal command.

AVAIL -- TYPE DISK USAGE STATISTICS

```
AVAIL [<packname>]
      [<currdisk>]
      [    *     ]      (All Started Disks)
      [   ONE    ]      (Logical disk #1)
          ...
      [SEVENTEEN ]
```

External command.

BASIC

```
BASIC [<filename>]
```

See BASIC Interpretive Language User Guide, MAN1813.

External command.

BASICV -- VIRTUAL MEMORY BASIC

```
BASICV [<filename>]
```

See the BASIC/VM Guide.

External command.

BASINP -- READ PAPER TAPE

```
BASINP <filename>
```

External command.

BINARY -- OPEN FILE UNIT 3 FOR BINARY OUTPUT

```
Binary <filename>
```

Internal command.

CHAP -- CHANGE USER PRIORITY

```
CHap -<usrno> [<level>] [<timeslice>]
        ALL   [   1   ] [     3     ]
```

<timeslice> is in tenths-of-a-second. Defaults taken only for ALL option, else unchanged.

System user only.
Internal command.

CLOSE -- CLOSE FILE UNIT(S)

```
Close <funit>...
      <filename>
          ALL
```

Unit '21 (COMOUTPUT) must be closed explicitly.

Internal command.

CMPF -- COMPARE ASCII FILES

```
CMPF <tree1> <tree2> [... <tree5>] [<option>...]
```

Options can be:

```
        -MINL [<n>]    (default = 3)
        -BRief
        -REPORT <report-file-name>
```

External command.

CMPRES -- COMPRESS SOURCE FILE

```
CMPRES <intreename> [<outtreename>]
                    [ <intreename>]
```

External command.

CNAME -- CHANGE NAME OF FILE

```
CName <oldfilename> <newfilename>
```

Internal command.

CNVIMA -- CONVERT LOAD MAP FOR PMA

```
CNVIMA <in-file> <out-file>
```

Converts load map into format usable by PSD 'LS' command.

External command.

COBOL

```
COBOL <treename> [<option>...]

     or

COBOL [<option>...] -I <treename> [...<option>]

Options can be:
  -Binary
     Define binary file/device.
  -B <treename>
     Create binary file with specified treename.
  -B NO
     Do not create a binary file.
  -B YES
     Create binary file in current UFD.
  -EXPlist
     Generate an expanded listing file.
  -Input
     Define input file/device.
  -I <treename>
     <treename> is source program.
  -Listing
     Define listing file.
  -L <treename>
     <treename> is listing file.
  -L NO
     Do not create a listing file.
  -L YES
     Create listing file in current UFD.
  -L TTY
     Print listing at terminal.
  -L SPOOL
     Spool listing file to line printer.
  -NOEXPLIST
     Do not generate expanded listing file.
  -64R
     Generate relative-addressed code.
  -64V
     Generate segmented-addresssed code.
```

External command.

COMINP -- START COMMAND INPUT FILE

```
COminp <filename> [<ufdname>] [<funit>]
       [-]PAUSE               [  6   ]
       [-]CONTIN
       [-]TTY
       -Start
       -End
```

'-S' = 'S' + 'CO CONTIN'. '-E' = 'TTY'.

Internal command.

COMOUTPUT -- CONTROL ROUTING OF TERMINAL OUTPUT

```
COMOutput [<filenamel>] <option>...

Options can be:
          -Ntty      -Contin
          -Tty       -Pause
          -End
```

Internal command.

X.CONCAT -- CONCATENATE FILES

```
CONCAT [<outtreename>]
```

No name => unit 2 assumed open.

Follow with list of files to be cancatenated together. '=' starts line to be used as header. Null line terminates list (unit 2 left open if open on entry).

External command.

COPY -- COPY DISK

```
COPY

Prompts:

FROM PHYSICAL DISK=  (Enter DVNO-see Disk Addresses.)
1.5M WORD PACK?   (Enter Yes or No.)
TO PHYSICAL DISK=  (Enter DVNO)
1.5M WORD PACK?   (Answer as to FROM...)
FROM,TO,RECORDS= <from-dvno> <to-dvno> <rec-in-dec>
PARAMETERS OK?  (Yes or No. No => reenter all.)

External command.
```

CPMPC -- PUNCH FILE ON CARD PUNCH

```
CPMPC <treename> [<option>...]

Options can be:
     -PRINT
     -CR0
     -CR1

External command.
```

CREATE -- CREATE SUBUFD IN CURRENT UFD

```
CReate <ufdname>

Internal command.
```

CRMPC -- READ CARDS

```
CRMPC <treename> [<option>...]

Options can be:
     -PRINT
     -CR0
     -CR1

External command.
```

CRSER -- READ FROM SERIAL CARD READER

```
CRSER <treename>

External command.
```

CX -- SEQUENTIAL JOB MONITOR

```
CX [<filename>] [<option>...]

Options can be:
  -A
     list entire audit file.
  -Dxx
     drop job numbered xx.
  -ON <dvno>
     submit CX job on <dvno> specified.
  -P
     list personal jobs in queue and audit file.
  -Q
     list queue.
  -Sxx
     list status of job numbered xx.

External command.
```

DATE -- PRINT DATE AND TIME

```
DATE

External command.
```

DELAY -- SET TERMINAL DELAY CHARACTERISTICS

```
DELAY [<min>] [<max>] [<width>]
      [  6  ] [ 12  ] [  72  ]

Can issue prior to login.
Internal command.
```

DELETE -- DELETE FILE

```
DELETE <filename>

Internal command.
```

DELSEG -- DELETE SEGMENT(S)

```
DELSeg <segno>
         ALL

segno > '2000 and not '6000

Internal command.
```

DISKS -- SPECIFY ASSIGNABLE DISKS

```
DIsks [NOT] <dvno> ...

System user only.
Internal command.
```

ED -- EDITOR

```
ED [<filename>]   (no <filename> => new file)

(<str> - text string)
(/ = unique delimiter not in string)
```

SUBCOMMANDS

```
 .CR.  = INPUT TTY
 Append <str>
    Append to current line.
 Bottom
    Go to bottom of file.
 BRief
    Don't display changes.
 Change/<str1>/<str2>[/] [<n>] [G]
    Change <str1> to <str2> for first occurrence  on
    line,  for all occurrences if G present, for <n>
    lines if <n> present.
 Delete [<n>]
    Delete <n> (1) lines.
 Delete TO <str>
    Delete to line containing <str>.
 DUnload <fname> [<n>]
    Unload/delete <n> (1) lines.
 DUload <fname> TO <str>
    Unload/delete up to (not incl)  line  containing
    <str> to <fname>.
 Erase <char>
    Make <char> the erase character (").
 FILe [<fname>]
    Write updated file to <fname>.
 Find <str>
    Find line starting with <str>.
 Gmodify <subcmnds>
    Modify line w/subcommands:
         A/<str>/ - Append
         B<n> - Back <n> chars
         C<c> - Copy up to (not inc) char <c>
         D<c> - Delete up to char <c>
         E<n> - Delete next <n> chars
         F - Copy to end of line
         I/<str>/ - Insert <str> at curr pos.
         M<n> - Copy <n> chars
         Nxx - Negate criteria of cmnd xx
         O/<str>/ - Overlay at current position
         R/<str>/ - Retype at current position
         S - Reset to start of line
```

```
Insert <str>
   Insert line (.NULL.  => input mode).
INPUT [ASR] [PTR] [TTY]
   Input text from specified device.
Kill <char>
   Make <char> new kill character.
LINesz <n>
   Set max line size to <n> chars.
LOAd <fname>
   Insert contents of <fname>.
Locate <str>
   Locate line containing <str>.
MODE <arg>
   Set editor mode.  <arg> can be:
        PRUPPER, PRALL, PRLOWER,
        PROMPT, NPROMPT,
        COUNT, NCOUNT,
        NUMBER, NNUMBER,
        COLUMN, NCOLUMN.
Modify /<str1>/<str2>[/] [G] [<n>]
   Copy <str2> on top of <str1> starting with first
   char.
MOVe <buf1> <buf2>
   Move contents of <buf2> to <buf1>.
MOVe <buf1> <str>
   Move contents of <str> to <buf1>.   Buffers  are
   EDLIN  (command line), INLIN (current line to be
   editted), STR.1, ..., STR.1Ø.
Next [<n>]
   Advance <n> (1) lines.
NFind <str>
   Find line not starting with <str>.
OUTput [DISPLAY] [TTY]
   Send verification output to specified device.
Overlay <str>
   Overlay line with <str>.  Blank  leaves  current
   char, WILD becomes blank.
PAuse
   Back to PRIMOS, restart w/'S'.
POint <n>
   Go to line <n>.
Print [<n>]
   Print <n> (1) lines
PSymbol
   Print symbols.
PUnch [<n>] [ASR] [PTP]
   Punch n lines on indicated device.
```

```
Quit
   Exit without filing.
Retype <str>
   Replace line with <str>.
Symbol <name> <char>
   Define <name> symbol. <name>: BLANK (#),
   CPROMPT ($), COUNTER (@), DPROMPT (&), ERASE
   ("), ESCAPE (^), KILL (?), SEMICO (;), TAB (\),
   WILD (!).
TAbset <tabl>...
   Set tab positions.
Top
   Go to top of file.
Unload <fname> [<n>]
   Unload <n> line into <fname>.
Unload <fname> TO <str>
   Unload lines up to (but not incl) <str> to
   <fname>.
Verify
   Display all changed lines.
Where
   Print current line number.
Xeq <buff>
   Execute contents of buffer.
*[<n>]
   Repeat <n> (until bottom or forever) times.
```

External command.

EDB -- BINARY EDITOR

```
EDB <intreename> [<outtreename>]
        PTR            PTR
```

SUBCOMMANDS

```
BRIEF
   No names printed.
Copy <name>
   Copies up to (but not incl) <name>.
Copy ALL
   Copies to end of file.
Find <name>
   Position to <name>.
Find ALL
   Position to end of file.
Insert <treename>
   Insert <treename>.
Newinf <name>
   Open new input file.
OPEN <name>
   Open output file.
Replac
   Replace <fname> with <treename>.
RFL
   Reset Force Load flag.
SFL
   Set Force Load flag.
TERSE
   Print 1st name in blocks.
Top
   Top of input file.
VERIFY
   Print all names.
```

To replace <name>:

```
EDB <oldlib> <newlib>
R <name> <treename>
C ALL
ET
QUIT
```

External command.

ELIGTS -- SET ELIGIBILITY TIMESLICE

```
ELIGTS <tenths>
          3
```

Internal command.

EXPAND -- EXPAND (COMPRESSED) SOURCE FILE

```
EXPAND <intreename> <outtreename>
```

External command.

FILMEM -- ZERO MEMORY

```
FILMEM [ALL]
```

No 'ALL' => '1ØØ - '77777 excluding DOS zeroed.

External command.

FILVER -- COMPARE BINARY FILES

```
FILVER [<treename1>] [<treename2>]
```

Prompts if no names entered.

External command.

FIXRAT -- FIX RECORD AVAILABILITY TABLE

```
FIXRAT [<option>...]
```

Prompts:

```
FIX DISK?   (Enter Yes or No.)
PHYSICAL DISK DRIVE= (Enter DVNO-see Disk Addresses.)
If 'OPTIONS' specified:
   TYPE DIRECTORIES TO LEVEL  (Enter level.)
   TYPE FILENAMES  (Enter Yes or No.)
   TYPE FILE CHAINS   (Enter Yes for disk addrs.)
```

External command.

FTN -- FORTRAN

```
FTN [-Input] <itree> [-B    <btree>] [-L    <ltree>]
                     [-B B<-<itree>] [-L L<-<itree>]
    [<options>] [<a-reg>] [<b-reg>]
                [ 1/17Ø7] [  2/Ø  ]
```

Option	Function
-64v	Generate 64V mode object code
-64R	Generate 64R mode object code
-32r	Generate 32R mode object code
-Big	Dummy arrays may cross segment boundaries
-NOBig	
-DClvar	Flag undeclared variables
-NODclvar	
-Debase	Conserve loader base areas
-DYnm	allocate locxal storage in stack frame
-ERRlist	Generate errors-only listing
-ERRTty	Print errors on terminal
-NOErrtty	
-Explist	Generate expanded listing
-Fp	Generate floating-pt skip instr
-NOFp	
-Intl	INTEGER*4 default
-INTS	INTEGER*2 default
-SAve	Allocate local storage in linkage frame
-Spo	System programmer option
-Trace	Generate code for trace output
-Notrace	
-XREFL	Generate cross reference listing
-Xrefs	Generate abbreviated cross ref listing
-NOXref	

A-REG		ON-OPTION	OFF-OPTION
1	100000	Spo	--
2	040000	Explist	ERlist
3	020000	ERlist	Explist
4	010000	Trace	Notrace
5	004000	64r	32R,64V,DYnm,Big
6	002000	Debase	--
7	001000	ERRTty	NOErrtty
8-10	000700	Input device (see below)	
11-13	000070	Explist,Erlist,Xrefs,XREFL (listing)	
14-16	000007	Binary device	--

B-REG

1	100000	(Debug triad dump)	--
2	040000	Unused	--
3	020000	SAve,DYnm	--
4-7	017000	Unused	--
8	000400	64V,DYnm,Big	64r,32r,Debase
9	000200	Big	NOBig
10	000100	Intl	INTS
11	000040	Unused	--
12	000020	Xrefs	XREFL,NOXref
13	000010	Xrefs,XREFL	NOXref
14	000004	Unused	--
15	000002	NOFp	Fp
16	000001	Spo,DClvar	NODclvar

Device codes for Input, Listing, Binary:

0 - None
1 - ASR
2 - PTR/PTP
3 - Card reader
4 - Line Printer
5 - Magtape
6 - Cassette
7 - Disk

External command.

FUTIL -- FILE SYSTEM UTILITY

```
FUTIL

N.B.: command names must be in upper case.

Attach <treename>    ('*' => home ufd)
CLEAN <prefix> [<level>]
Copy <file> [<newname>] [,<file>[<newname>]] ...
COPYSam <file> [<newname>] [,<file>[<newname>]] ...
COPYDam <file> [<newname>] [,<file>[<newname>]] ...
CReate <ufdname> [<owner> [<non-owner>]]
DELETE <file> [,<file>] ...
FOrce ON or OFF
From <treename>    ('*' => home ufd)
Listf [<level>] [First] [LISTFIL] [PROtect] [Size]
      [Type] [Date] [Rwlock] [PAssw]
LISTSave <filename> [<options as for Listf>]
Protect <file> [<owner>] [<non-owner>]
Quit
Scan <file> [<options as for Listf>]
SRwloc <file> <lockno>
To <treename>
TRECpy <ufd> [<newname>] [,<ufdname>[<newname>]]...
TREDEL <ufdname> [,<ufdname>] ...
TREPro <ufdname> [<owner> [<non-owner>]]
TRESrw <ufdname> <lockno>
Ufdcpy
UFDDEL
UFDPro [<owner> [<non-owner>]]
UFDSrw <lockno> <level>
```

<lockno>:

0	use system read/write lock (SYS)
1	n readers or 1 writer (W/NR)
2	n readers and 1 writer (1WNR)
3	n readers and n writer (NWNR)

External command.

HPSD -- HIGH PSD

HPSD

SA, EA = 147760, 156552.
Start of initial P counter = 150000.
For internal commands, see PSD.

External command.

INPUT -- OPEN FILE UNIT 1 FOR INPUT

```
Input <filename>
```

Internal command

LISTF -- LIST FILES IN CURRENT UFD

```
Listf
```

(Note: LISTING command with no name => LISTF.)
Internal command.

LISTING -- OPEN FILE UNIT 2 FOR LISTING OUTPUT

```
Listing <filename>
```

(Note: <filename> omitted => LISTF.)
Internal command.

LOAD

```
LOAD

ATtach [<ufd>] [<password>] [<ldisk>] [<key>]
          <key>=0=>don't set home, 1=>set home.
AUtomatic [<n>]     Linkareas of length <n> around
          module.  <n>=0 turns feature off.
CHeck [<symbol>] [<par1 ... par9>]
COmmon <address>      Set COMMON TOP - 1
DC [END]
ENtire <treename>
ERror [<num>];  <num> = 0, 1, or 2
EXecute [<a>] [<b>] [<x>]    Uses START entry
FOrceload <treename> [<addr>] [<linkstart>]
          [<linkrange>]
F/        Force prefix for FO, LO, LI commands.
HArdware <definition>
          177700 Must be zero
          000040 1=>Prime 400 instruction set
          000020 Unused
          000010 1=>Double prec.  fl.  pt.
          000004 1=>Single prec.  fl.  pt.
          000002 1=>Prime 300 instruction set
          000001 1=>High speed arithmetic
INitialize <treename> [<addr>] [<linkstart>]
          [<linklen>]
          Resets everything and loads <treename>.
LIbrary [<treename>] [<addr>]
          [LIB>FTNLIB]
LOad <treename> [<barea1>]...[<barea8>]
   <treename> [<barea1>]...[<barea9>]
   <treename> <symbol> [<barea1>]...[<barea9>]
```

```
MAp [<treename>] [<option>]
    [   $F   ]  <option> = 0=>full map, 1=>load state,
          2=>load state and link info, 3=>unresolved
          references, 4=>same as 0, 5=>system
          programmer map, 6=>sorted unresolved
          references, 7=>sorted full map, 10=>symbol
          map for PSD.
MOde [D32R] [D64R] [D16S] [D32S] [D64V] [D32I]
P/ Page boundary prefix for FO, LO, LI commands.
PAuse
PBrk [<symbol>] [<par1 ...  par9>]
         *       <par1> [<par2 ... par9>]
QUit      Back to PRIMOS
SAve <fname> [<a>] [<b>] [<x>] [<keys>]
SEtbase [<linkstart>] [<linklen>]
         *      [end of sector] (*=>current sector)
SS <symbol>
SYmbol <symbol> <oldsym> [<par1 ...  par6>]
   <symbol> <addr> [<par2 ...  par6>]
   <symbol>    *   [<par1 ...  par3>]
          parameters can contain + and - signs
SZ [NO] or SZ YES
VIrtualbase <linkstart> <tosector>
XPunge [<y>] [<z>]      <y>:  0=>all but undefined
          symbols, 1=>all but undefined and COMMON.
          <z>:  0=>all defined base areas, 1=>all but
          sector 0, 2=>return all.
```

External command.

LOGIN -- LOGIN TO UFD

```
LOGIN <ufdname> <passwd> [<dvno>] [-ON <nodename>]
```

Internal command.

LOGOUT -- LOGOUT USER

```
LOgout [-<usrno>]
          ALL       (System user only)
```

<usrno> must have same login name as user unless issued by System user.

Internal command.

LOGPRT -- PRINT LOGREC

```
LOGPRT [<outfile>] [<option>...]
       [ LOGLST ] [ -Help ]
       [   Tty  ] [ -From <mmddyy> ]
                  [ -Type Cold
                          Warm
                          Timdat
                          CHecks
                          Disk
                          DSKnam
                          Overfl
                          Shutdn
                          CHK300
                          Par300
                          Mod300
                          TYPE10-TYPE15 ]
                  [ -Spool  ]
                  [ -Delete ]
                  [ -PURGE  ]
```

Prompts for input treename, default (just .CR.) is CMDNCO>LOGREC.

External command.

LOOK -- MAP SEGMENT TO USER 1

```
LOOk [-<usrno>] [<segno>] [<access>] [<mapseg>]
     [-   1   ] [  6000 ] [   200  ] [  4001  ]
```

System user only.
OPRPRI 1.
Internal command.

X.LS -- LIST FILE CHARACTERISTICS

```
LS [<filename>] [<option>]...
```

Filename can be a wild card name containing % and + signs:

% matches any number of characters (including none) in the filename;

+ matches any one character in the filename.

Options can be:

```
-DTM        -TYPE
-PR         -RWLK
```

External command.

MACHK -- TURNS ON MACHINE CHECK MODE

```
MACHK
```

External command.

MAGNET -- TRANSFER DATA TO AND FROM TAPE

MAGNET

PROMPTS	RESPONSES
OPTION:	POSITION or READ or WRITE or COPY

Depending on the option, MAGNET may prompt any of the following --

MTU # =	#/7 or #/9; Ø≤#≤7
ABSOLUTE OR RELATIVE?	A or R
FILE # =	if # if A mode, #≥1; if R mode, # can be either positive or negative

RECORD # =	if A mode, #≥1;
	if R mode, #≥Ø

MT FILE # =	#≥Ø
LOGICAL RECORD SIZE =	≤ 1ØK bytes for PRIMOS II (DOS), IV, V or ≤ 2K bytes for PRIMOS III
BLOCKING FACTOR =	# of line images in one tape record
ASCII, BCD, BINARY or EBCDIC?	ASCII or EBCDIC or BCD or BINARY
FULL OR PARTIAL RECORD TRANSLATION?	FULL or PARTIAL
OUTPUT FILE NAME:	<filename>
INPUT FILE NAME:	<filename>
STARTING FILE # =	
# FILES TO COPY =	

External command.

MAGRST -- MAGTAPE RESTORE

```
MAGRST [-7TRK]
        -9TRK
```

PROMPTS	RESPONSES
TAPE UNIT:	Ø - 7
ENTER LOGICAL TAPE NUMBER:	Ø -- tape already positioned 1 -- first logical tape 2 -- second logical tape etc.
READY TO RESTORE:	Yes (yes) No (no) PA (partial) $I [<filename>] [<level>] (turn on indexing) NW [<level>] (index only)
TREE NAME:	<treename> (often partial) or list of treenames, 1 per line, end with null line

External command.

MAGSAV -- MAGTAPE SAVE

MAGSAV [<option>...]

Options can be:

- -LONG
  1Ø24 word records (default is 512).
- -7TRK
  use 7 track tape format (default is 9 track).
- -INC
  incremental save (only save files that have been modified since last save).
- -UPDT
  set dumped bit in the UFD entry (default is not to set the dumped bit).

PROMPTS	RESPONSES
TAPE UNIT:	Ø -7
ENTER LOGICAL TAPE NUMBER:	Ø -- tape already positioned 1 -- first logical tape 2 -- second logical tape etc.
TAPE NAME:	<6-character name>
DATE:	MM DD YY or .CR. for today's date (under PRIMOS III and IV)
REV NO:	<an arbitrary integer>
NAME:	<filename> $A [<UFDname>] (attach) $Q (terminate tape and return to PRIMOS) $R (terminate tape, rewind, and return to PRIMOS) $I [<filename>] [<level>] (print index to indicated level) MFD (save entire disk) * (save current directory)

External command.

X.MAIL

MAIL [<ufdname>]

If no <ufdname> is specified, the mail command checks mail for the user issuing the command.

When sending mail to another user, end message with '$'.

External command.

MAKE -- FORMAT DISK

```
MAKE [OLD]
     [NEW]
```

PROMPT	RESPONSES
PHYSICAL DISK	physical disk number
1.5 WORD PACK?	yes or no
SPLIT DISK?	yes or no
PAGING RECORDS (DECIMAL)	number of records to be used for paging (see paging records tables, below)
<DISK NUMBER> <FILE RECORDS> <PAGING RECORDS> OK?	yes or no
BADSPOTS ON DISK?	yes or no
TRACK=	track of badspot or Ø to terminate
HEAD=	head of badspot or 9 to terminate
prints list of badspot HEAD and TRACK numbers: PARAMETERS OK?	yes or no
VIRGIN DISK?	yes or no
VERIFY DISK?	yes or no

PAGING RECORDS TABLE

Disk	Decimal Records
Diskette	46Ø
1.5 million word disk	3248
3.Ø million word pack	6496
3Ø million word disk	6496Ø
128 K word fixed head disk (32 trk)	256
256 K word fixed head disk (64 trk)	524
512 K word fixed head disk (128 tk)	1Ø24
1Ø25 K word fixed head disk (256 tk)	4Ø96

RECORDS PARAMETERS FOR 3Ø-MILLION WORD DISK

Partition	Dev Addr 23 Disk Number	Dev Addr 21 Disk Number	Records
2 hd (deflt)	XXØ25X	XXØØ5X	6496
2 hd (explicit)	XXØ65X	XXØ45X	6496
4 hd	XX125X	XX1Ø5X	12992
6 hd	XX165X	XX145X	19488
8 hd	XX225X	XX2Ø5X	25984
1Ø hd	XX265X	XX245X	3248Ø
12 hd	XX325X	XX3Ø5X	38976
14 hd	XX365X	XX345X	45472
16 hd	XX425X	XX4Ø5X	51968
18 hd	XX465X	XX445X	58464
2Ø hd	XX525X	XX5Ø5X	6496Ø

External command.

! MAXSCH -- SET SCHEDULING CONSTANT

```
MAXSch <n>
        3
```

System user only.
Internal command.

! MAXUSR -- LIMIT NUMBER LOGGED-IN USERS

```
MAxusr [<number>]
       [  64   ]
```

System user only.
Internal command.

MDL -- MEMORY DUMP/LOAD

```
MDL
```

MDL command requests parameters. Respond with the following data separated by spaces or commas and terminate with a CR or LF:

PARAMETER	DEFINITION
SA	first location to be punched, must be ≥ '34.
EA	last location to be punched, must be ≤ '177777.
P	auto-start address or Ø.
K	keys (default is 16S mode). bit 14: Ø punch end of tape (EOT). 1 omit EOT. bit 15: Ø punch begin of tape (BOT). 1 omit BOT. bit 16: Ø high speed punch. 1 ASR punch
L	loader address (default is sector in which MDL resides); if ASR, avoid folowing loader address: ' 1Ø4ØØ thru ' 116ØØ '11Ø4ØØ thru '1116ØØ

Internal command.

MESSAGE -- SEND MESSAGE TO USER(S) OR SYSTEM

```
Message -<usrno> [NOW]
          [-1]
          ALL
          NET
          <nodename>
```

End command with .CR., enter message on next line, end message with .CR.

System user for all but 'M' (user to operator).
Internal command.

MRGF -- MERGE ASCII FILES

```
MRGF <treel> <tree2> [...<tree5>] outtree [<opts>]

Options can be:
       -MINL [<n>]   (default = 3)
       -BRief
       -FORCE
       -REPORT <report-file-name>
```

External command.

NUMBER -- (RE)NUMBER BASIC FILE

```
NUMBER
```

External command.

OPEN -- OPEN FILE ON SPECIFIED UNIT

```
Open [<filename>] <unit> <key>
```

<key>: 1-Read, 2-Write, 3-R/W, 4-Close, 5-Delete, 6-Exist, 7-Rewind, 1Ø-Truncate + ØØØØ-New SAM, 2ØØØ-New DAM, 4ØØØ-New SAM segment, 6ØØØ-New DAM segment, 1ØØØØ-New UFD. <filename> optional only for Rewind and Truncate.

Internal command.

OPRPRI -- SET OPERATOR PRIORITY

```
OPRpri  [ 1 ]
        [ Ø ]
```

System user only.
Internal command.

PASSWD -- SET PASSWORDS ON CURRENT UFD

```
PASSWD [<owner-password>] [<non-owner-password>]
       [     blanks     ] [       blanks       ]
```

Internal command.

PHANTOM -- START PHANTOM USER

```
PHantom <filename> [<funit>]
                   [   6   ]
```

Phantom file should end with 'CO TTY' command.

Internal command.

PM -- PRINT USER REGISTER VECTOR

```
Pm
```

Internal command.

PMA -- PRIME MACRO ASSEMBLER

```
PMA [-Input] <itree> [-B    <btree>] [-L    <ltree>]
                     [-B B<-<itree>] [-L L<-<itree>]
    [<options>] [<a-reg>] [<b-reg>] [<x-reg>]
                [ 1/777 ] [  2/Ø  ] [  3/Ø  ]
```

External command.

Options:

Errlist	Errors-only listing
EXplist	Expanded listing

A-REG		ON-OPTION	OFF-OPTION
1	1ØØØØØ	Unused	--
2	Ø4ØØØØ	Errlist	EXplist
3	Ø2ØØØØ	EXplist	Errlist
4-7	Ø17ØØØ	Unused	—
8-1Ø	ØØØ7ØØ	Input device (see below)	
11-13	ØØØØ7Ø	Errlist,EXplist (normal listing)	
14-17	ØØØØØ7	Binary device	—

Device codes (Input, Listing Binary):

Ø - None	4 - Line Printer
1 - ASR	5 - Magtape
2 - PTR/PTP	6 - Cassette
3 - Card Reader	7 - Disk

B-REG (PRIMOS IV BUILD):

11-13	ØØØØ2Ø	64-user version
	ØØØØØØ	16-user version
16	ØØØØØ1	Large 16-user version

PMA ERROR CODES:

C: INST IMPROPERLY TERMINATED.

F: BAD MACRO EXPR TERMINATOR, ILLEGAL OP ON STACK PUSH/POP, FAIL PSEUDO-OP.

G: GOTO ERROR WITHIN MACRO, END/ENDM IN 'GOTO' SKIP AREA.

I: GENERIC, I/O, OR SHIFT HAS TAG, TAG ON 32I FIELD INSTR, SHORT INSTR SPEC (#) IMPOSSIBLE, 64V LDX CLASS BAD TAG, 64V TAG ON BRANCH ILLEGAL, SEG EXT REF BAD INDIRECT OR INDEX, AP/IP, INDEX SPECIFIER INVALID, TAG ON 32I BRANCH.

L: BAD LABEL, EXTERNAL VARIABLE IN LITERAL, BAD ARG IN EQU. SET, OR XSET,

M: MULTIPLY DEFINED LABEL.

N: 'END' WITHIN MACRO OR IF.

O: BAD OPCODE, 64V MEMORY REFERENCE WHEN NOT IN 64V MODE, S/R MODE MEMORY REFERENCE NOT IN S/R MODE.

P: MISMATCHED PARENTHESIS.

Q: AP, NOT IN 64V/32I MODE, IP, NOT IN 64V/32I MODE, ENDM PSEUDO-OP NOT IN MACRO.

R: STACK OVERFLOW, MULT DEF MACRO OR NAME EMPTY.

S: 'LOAD' MODE, INSTR NEEDS DSECT, INDIRECT DAC IN C64R MODE.

T: 32I MODE TAG MOD SYNTAX ERROR.

U: UNDEF VAR IN ADR FIELD/EXP, UNDEFINED VARIABLE IN ORG/SETB.

V: BIT FIELD OUT OF RANGE, UNRECOGNIZED OPERATOR IN EXPRESSION, FIELD ADR INST, OUT OF RANGE, I/O INSTR FIELD OUT OF RANGE, INST, SHIFT COUNT OUT OF RANGE, NO COMMA FOLLOWING FAR SPEC, 32I NO COMMA AFTER REGISTER #, 32I NO COMMA AFTER BIT #, 32I BAD DELIMITER, 32I SHIFT INSTR, BAD DELIM, BAD COUNT IN 32I SHIFT INSTR, BAD TAG MODIFIER IN 32I SHIFT, BAD DELIM AFTER REG # 32I PIO, OPEN PAREN MISSING ON DFTB ARG, CLOSE PAREN MISSING ON DFTB ARG, NO LAB IFTF, IFTT, IFVT, IFVF, NO NAME IFTF, IFTT, IFVT, IFVF, ABS/REL ILLEGAL IN SEG MODE, SEG/SEGR AFTER CODE GENED, PROC/LINK OUTSIDE OF SEG MODE, FIELD OUT OF RANGE ON DDM, BAD ARGUMENT FOLLOWING 'EXT', 'END' WITHIN MACRO, SYNTAX ERR IN 'DYNM' PSEUDO-OP, BAD ARG ON SUBR STATEMENT, VFD PSEUDO-OP, 16 BITS NOT DEFINED, UNTERMINATED CHARACTER STRING, EXPRESSION OVERFLOW ON FL PT NORMALIZE, EXPRESSION OVERFLOW ON FL PT RE-NORMALIZE, SCALED BINARY LOSS OF SIG, FL POINT NUMBER OUT OF RANGE, BCI REPEAT COUNT ERROR, BCI COUNT VARIABLE TYPE ERROR, MUNG IN ADDR FIELD OF CALL, BAD ADDR FIELD ON COMN, REPEAT COUNT ERROR, DEC/OCT PSEUDO OP HAS BAD OP, RLIT FOUND AFTER CODE GENED, NO LABEL ON DFTB.

X: 32I MODE GPR SPEC ERROR.

Y: PHASE ERROR.

Z: ILLEGAL ABS REF IN SEG MODE, SEG MODE ABS REF NOT Ø-7, AP/IP, ABSOLUTE REF INVALID, TOO MANY EXT NAMES IN EXPR, BAD EXPR MODE FOR INSTR, EXPRESSION MODE ERROR, >1 NON-ABS/REL OPERATOR, RIGHT-HAND OP NOT ABS/REL, EXTERNAL NAME NOT PERMITTED.

PRERR -- PRINT ERRVEC AND LAST ERROR MESSAGE

PRerr

Internal command.

PRMPC -- PRINT FILE ON LINE PRINTER

PRMPC <treename>

External command.

PROTECT -- SET PROTECTION ON FILE

```
PROtec [<owner-rights>] [<non-owner-rights>]
       [      Ø       ] [        Ø         ]
```

Ø-No access, 1-Read, 2-Write, 3-R/W, 4-Delete/Truncate, 5-D/T/R, 6-D/T/W, 7-All. Default on file creation equals 7 Ø.

Internal command.

PRSER -- PRINT FILE ON SERIAL LINE PRINTER

PRSER <treename>

External command.

PRVER -- PRINT FILE ON VERSATEC

PRVER <treename>

External command.

PSD -- PRIME SYMBOLIC DEBUGGER

PSD [<token>...]

(NOTE: VPSD has: segment, base register operations, does not have: symbols, trace.)

TERMINATORS for 'A'		MODES	
.CR.	*+1	:A	ASCII
,	*+1	:B	BINARY
^	*-1 (uparrow)	:H	HEXIDECIMAL
.n	*+n	:O	OCTAL
.-n	*-n	:S	SYMBOLIC
@	Effective address	:D	DECIMAL
\	Back to last @	:P	AP
(	To contents of *	:L	LONG OCTAL INTEGER

```
)    Back to last defined (
=    EA + contents, no update of *
/    Return, do not close *
?    return, do not close *
!    Return, close *
```

Expressions: Locations can be expressions including:

```
* (current location)
[+]number-in-current-mode
>number-relative to relocation constant
```

SUBCOMMANDS

```
Access <loc>
   Access location.
Breakpoint <loc>
   Set breakpoint (up to 1Ø).
BR
   Print base registers.
Copy <from> <to> <new-addr>
   Copy block of memory to new location.
Define <sym> <val>
   Define symmbol.
Dump <from> <to> [<ncol>] [<mode>]
   Dump contents of memory.
Effective <from> <to> <match> [<mask>]
   Search for effective address.
Execute <addr> [<a>] [<b>] [<x>]
   Search for effective address.
EXecute
   Execute segmented program.
FAddress <fld-addr-reg-no>
   Access field address register.
FLength <fld-len-reg-no>
   Access field address register.
Fill <from> <to> <val>
   Fill memory block with <val>.
GO [<count>] [<a>] [<b>] [<x>] [<k>]
   Continue at breakpoint.
Jumptrace [<start>] [<a>] [<b>]
   Execute obj prog and produce diagnostic listing.
Keys <value>
   Set keys to value.
List <loc>
   list location.
LB <sn> <wn>
   Set link base.
LS
   Load symbols (unit 1).
```

```
MAp
   Print load map symbols.
MO [D16S] [D32R] [D64R] [D64V] [D32S] [D32I]
   Set address mode.
Monitor [<start>] [<a>] [<b>] <addr>
   Trace obj prog for mem ref instr.
Not-equal <from> <to> <nmatch> [<mask>]
   Negative serarch.
Open <fname> <unit> <key>
   Open unit.
PATCH <locl> <loc2>
   Patch instr  in <loc2> into <locl>.
Print
   Print brkpt, contents, a, b, x,
   keys, relocation.
PRoceed [<newbrk>] [<a>] [<b>] [<x>] [<k>]
   Set new brkpt and resume execution.
Quit
   Quit.
RElocate <reloc-val>
   Set relocation constant.
Run [<loc>] [<a>][<b>][<x>][<keys>]
   Run program.
SB <sn> <wn>
   Set stack base.
Search <from> <to> <match> [<mask>]
   Search memory block.
SN <sn>
   Set segment number.
SY Ø
   Symbol mode off.
SY 1
   Symbol mode on.
Trace [<addr>] [<a>] [<b>] [<val>] [-1 <interval>]
   Trace program.
Update <loc> <val>
   Update location.
Verify <from> <to> <copy-addr>
   Verify block of memory.
VErsion
   Print version, restart address.
Where
   Display brkpts and proceed counts.
X <reloc-val>
   Set relocation constant.
XB <sn> <wn>
   Set X base.
XR <val>
   Set X register.
YR <val>
   Set Y register.
Zero [<brk-loc>]
   Remove brkpt (current)
```

External command.

PTBOOT -- PAPER TAPE BOOT

```
PTBOOT
```

External command.

PTCPY -- PAPER TAPE COPY

```
PTCPY
```

External command.

PUSS -- COMPARE SOURCE FILES

```
PUSS
```

PROMPTS	RESPONSES
DIFF FILE, OMISSIONS?	name of difference file followed by a space plus YES, NO, or carriage return
OLD-FILE TREE-NAME:	<treename>
NEW-FILE TREE-NAME:	<treename>

Obsolete — use CMPF.
External command.

RESTORE -- RESTORE EXTERNAL PROGRAM

```
RESTore <filename> [<sa>] [<ea>] [<p>] [<a>]
   [<b>] [<x>] [<k>]
```

Internal command.

RESUME -- RUN EXTERNAL PROGRAM

```
REsume <filename> [<token>...] [<sa>] [<ea>] [<p>]
   [<a>] [<b>] [<x>] [<k>]
```

Internal command.

RPG -- REPORT PROGRAM GENERATOR

```
RPG <filename> [<option>...]
```

Options can be:

```
     -SEQCHK, -NOSEQCHK
     -BANNER, -NOBANNER
     -OBDATA, -NOOBDATA
     -STATUS, -NOSTATUS
     -ERRTTY, -NOERRTTY
     -LISTING
     -BINARY
```

Error Message Format:

```
****ERROR, LINE (xxxx) COL yy-zz [contents]-message.
**WARNING, LINE (xxxx) COL yy-zz [contents]-message.
```

xxxx = Line number of statement in error
yy-zz = Beginning and ending column number of
        the field in error. If yy is the same
        as zz, the zz portion is omitted
[contents] = contents of columns yy-zz
message = comments about the field in error

External command.

RUNOFF -- TEXT FORMATTER

```
RUNOFF [<filename>]
```

Notes:

1) When imbedded in text, all runoff command lines begin with a period; when issued at command level, runoff commands do not begin with a period.
2) In the table below, some runoff command actions are followed by brk, ejt, and/or deflt to indicate the command causes a break, ejects a page, and/or is the default. Also, if the runoff command has a default value, that value is specified.

(<str> = text string)

SUBCOMMANDS

```
.NULL.
   Start processing (from command mode).
* <str>
   Comment line.
+ <str>
   Enter verbatim string.
/-/-/-/
   /Left/Center/Right/ strings.
> <str>
   Center string.
Adjust
   Enter adjust/fill modes (brk, deflt).
BLank <char>
   Define blank substitute character (.NULL.).
BMargin <n>
   Set bottom margin (brk, ejt, 5).
Break
   Break (start new line).
CMargin <n>
   Set column margin (brk, ejt, 5).
Column
   Set number of columns (brk, ejt, 1).
DD <str>
   Down Decimal level.
DDS <str>
   Down decimal level, no decimal number.
Define <sym> <str>
   Define symbol value.
DI <lev> <before> <after>
   Set decimal indents.  Ø => all levels
DLevel <n>
   Go to decimal level <n> (1).
DLI <n>
   Set highest decimal level to appear in Table of
   Contents (all).
DN <str>
   Next heading on current decimal level.
```

```
DNS <str>
   Next heading on current decimal level,  suppress
   number.
DR <n>
   Reset number on decimal level <n>.
DS <lev> <before> <after>
   Set decimal heading skip values.
   Ø => all;  -1 => eject before
DU [<n>]
   Go up <n> decimal levels (1).
EFooter /-/-/-/
   Define even-page footer.
EHeader /-/-/-/
   Define even-page header.
Eject
   Page eject (brk, ejt).
ERase <char>
   Define cmnd mode erase char.
ERRgo
   Continue on error.
FILE <fn>
   Specify output file.
Fill
   Enter fill mode.
FLoat <fn>
   Floating insert of <fn>.
FOoter /-/-/-/
   Define footer for all pages.
FRom <n>
   First page number to output.
Header /-/-/-/
   Define header for all pages.
HYphen <char>
   Define phantom hyphen char (.RUBOUT.).
Indent <n>
   Indent left margin (5).
INDEX <str>
   Write <str> and page number to index.
INS <fn> [(<parms>]
   Insert <fn>.
INS <unit>
   Insert from <unit>.
IXfile <fn>
   Define index file (16).
Kill <char>
   Define command line kill char (?).
Length
   Specify physical page length (brk, ejt, 66).
NAdjust
   Leave adjust mode (brk).
Need <n>
   Eject if < <n> lines (1).
NERrgo
   Stop on error encountered (deflt).
```

```
NFill
   Leave fill and adjust modes (brk).
NFILE
   No output to file.
NIXfile
   Stop output to index file.
NParagraph
   No paragraph indentation (deflt).
NPAUse
   No pause between pages (deflt).
NPErforate
   No perforation marks (deflt).
NTty
   No output to TTY (deflt).
OFooter /-/-/-/
   Define odd-page footer.
OHeader /-/-/-/
   Define odd-page header.
PAGen <n>
   Set page number (1).
Paragraph [<n>] [<m>]
   Start paragraph, skip <n>, indent <m>.
PAUse
   Pause between output pages.
PErforate
   Print perforation marks.
PIcture <n>
   Leave <n> lines together (1).
PUrge
   Force in outstanding floats.
Quit
   Exit RUNOFF (brk, ejt).
RBar [ON]
   Start revision bars.
RBar [OFF]
   Stop revision bars.
REturn <n>
   Return to prev input file (Ø).
Rindent <n>
   Indent right margin (5).
RUndent <n>
   Undent right margin (Ø).
Skip <n>
   Skip <n> lines (brk, 1).
SM <n>
   Specify side margins (brk, ejt, 7).
SO <n>
   Print <n>th source line # (1).
Space <n>
   Specify single/double, etc.  spacing (1).
STop
   Conditional .QUIT/.RETURN.
SYchar <char>
   Define symbol delimiter (%).
```

```
Tab <char> <nl> ...
   Set tab character and stops.
TMargin <n>
   Specify top margin (brk, ejt, 7).
TO <n>
   Specify last page to print (32767).
TOFc <fn> <lim>
   Specify table of contents file.
TOFc [<opt>]
   Close, stop, start table of contents for
   <opt>=omitted, Ø, 1.
TTOfc <str>
   Enter string in table of contents.
TTy
   Output to TTY.
UNDEFine <sym>
   Undefine symbol.
Undent <n>
   Undent left margin.
WIDOw <n>
   Specify allowable widow size (Ø).
Width <n>
   Specify paper width (brk, ejt, 85).
```

External command.

SAVE -- SAVE MEMORY IMAGE

```
SAve <filename> [<sa>] [<ea>] [<p>] [<a>] [<b>]
   [<x>] [<k>]
```

Do not use SAve with 64V segmented files.

Internal command.

X.SAVER -- COMPARE RUNFILES

```
SAVER [<treename1>] [<treename2>]
```

Prompts if no names entered.

External command.

SEG -- SEGMENTED LOADER

```
SEG [<filename>]

SUBCOMMANDS
  DELETE [<filename>]
     deletes runfile.
  HElp
     print list of SEG commands.
  LOad [<treename>]
     define runfile and invoke loader for creation.
  LOad * [<treename>]
     define runfile and invoke loader for appending.
    ATtach [<UFDname>] [<password>] [<ldisk>] [<key>]
       attach to UFD.
    A/SYmbol <sname> [<segtype>] <segno> <size>
       define a symbol in memory and reserve space
       for it using absolute segment numbers.
    COmmon [ABS] <segno>
       relocate COMMON using absolute segment
       numbers.
    COmmon REL <segno>
       relocate COMMON using relative segment
       assignment.
    D/IL (D = Ditto = use parms of previous cmnd.)
    D/LOad
    D/LIbrary
    D/FOrceload
    D/PL or D/RL
       load using previous parameters.  D/ and F/ may
       be combined.
    EXecute
       save load to disk and execute program.
    F/xx [<filename>] [<addr psegno lsegno>]
       forceload all routines in object file.
    IL [addr psegno lsegno]
       load impure FORTRAN library.
    INitialize [<treename>]
       initialize and restart loader.
    LIbrary [<treename>] [addr psegno lsegno]
       load library file.
    LO [<treename>] [addr psegno lsegno]
       load object file.
```

```
    MAp [<filename>] <option>
       generate load map.
    MIxup [ON] or MIxup OFF
       mixes procedure and static data.
    MV
       moves portion of loaded file.  Will prompt for
       info.
    Operator
       relax/impose high-level restrictions
    PL [addr psegno lsegno]
       load pure FORTRAN library.
    P/xx [<filename>] [<option>] [<psegno
     lsegno>]
       load on a page boundary.
    QUit
       return to PRIMOS command level.
    REturn
       return to SEG command level.
    RL <treename> [<addr psegno lsegno>]
       reload a routine.
    R/SYmbol <sname> [<segtype>] <segno> <size>
       define symbol in memory and reserve space for
       relative segment assignment.
    SAve [a] [b] [x]
       save load to disk.
    SE <segno> <len>
       create base area for desectorization.
    Split <segno><addr>
        <addr>
        <addr><ssegno><saddr> <esegno>
       break data into data and procedure portions
    SS <sname>
       save symbol.
    STack <size>
       change stack size.
    SYmbol [<sname>] <segno> <addr>
       define a symbol at specific location in
       memory.
    S/xx [<filename>]  <addr> <psegno> <lsegno>
       load a specific absolute segment.
    XP <dsymbol> <dbase>
       expunge symbols from symbol table and delete
       base information.
  MAp <filename1> [<filename2>] <option>
               ascending addr, 11=sym>
    MOdify [<filename>] or SA [<filename>]
       invoke modification subprocessor.
     NEw <filename>
        write new copy of runfile to disk.
     PAtch
        modify save range of existing segment.
     REturn
        return to SEG command level.
     SK <ssize> or SK <segno> <addr>
        alter stack size and/or location.
     STart <segno> <addr>
        change program execution start address.
     WRite
        write all segments to disk.
   PArams [<filename>]
      display parameters of runfile.
   PSd
      invoke VPSD debugging utility.
   Quit
      return to PRIMOS command level.
   RESTore [<treename>]
      bring runfile into user memory.
   RESUme [<treename>]
      restore runfile and begin execution.
   SHare [<treename>]
      create R mode runfiles for segments below '4001.
   SIngle [<treename>] <segno>
      create R mode file image of single segment.
   TIme [<treename>]
      print time and date of last runfile
      modification.
   VERSION
      display SEG version number.
```

External command.

SETIME -- SET DATE AND TIME

SEtime -<mmddyy> -<hhmm>

Must be issued before user logins possible.
System user only.
Internal command.

SFRWLK -- SET FILE READ/WRITE LOCK

SFRWLK

Prompts for filename, RWLOCK setting (0-3): 0 - use system lock, 1 - N readers or one writer, 2 - N readers and one writer, 3 - N readers and N writers.

External command.

SHARE -- SPECIFY SHARED SEGMENT

```
SHAre [<filename>] <segno> [<access>]
                           [  600   ]
```

Omitted <filename> => change access only. <access>: 000-no access, 200-read access, 600-read/execute access, 700-read, write, execute access.
segno < '4000.

System user only.
OPRPRI 1 only.
Internal command.

SHUTDN -- SHUTDOWN DISK(S) OR SYSTEM

```
SHutdn [<node>] [<dvno> ... ]
                [  ALL ]
```

Do not shutdown logical device 0 -- contains CMDNC0.

System user only.
Internal command.

SIZE -- PRINT SIZE OF FILE

SIZE <treename>

External command.

SLIST -- PRINT FILE TO TERMINAL

SLIST [<treename>]

External command.

SORT

SORT [option1] [option2]

Options are: BRief, SPace, and MErge.

Prompts for: input filename, output filename, number of pairs of starting and ending columns, input pairs of starting and ending columns, reverse order, and data type. If merge option: number of files to be merged followed by input files, one per line.

External command.

SPOOL -- PRINT QUEUE MANAGER

```
SPOOL [ <treename> ] [<options>]
      [ %<treename>] [ -Lnum   ]  Line Numbers
                     [ -Ftn    ]  FTN Forms Ctl
                     [ -Expand ]  Expand
                     [ -Nofmt  ]  No Format Ctl
                     [ -Plot <nw> ]  Plot
                     [ -Defer [<time>]] Defer to time
                     [ -Home   ]  Local Printer
                     [ -FOrm <name> ]  Forms type
                     [ -Open   ]  Open only
                     [ -FUnit <unit> ] From unit
                     [ -Tunit <unit> ]  To unit (2)
      [   -CMD     ]
      [ -CANCEL <filename> ]
      [ -LIST [ ALL ] ]
              [ OWN ]
              [DEFER]
              [ PLOT]
              [PRINT]
              [FORM <type>]  (For default, say: ' ')
```

Prompts for defer time (-D), form type (-T) if not entered. '-CMD' ('SYSTEM' only) allows internal commands:

```
USER <usrno-of-spooler>
ABORT     (Abort current file and restart later)
DROP      (Aborts and deletes)
BACK      (Backs up 1ØØ lines and restarts)
RESTART   (Restart current file)
FORM <type>   (Specify new paper type)
HANG      (Stop now)
GO        (Restart after HANG, FINISH)
LENGTH <n>   (Specify paper length)
QUIT      (Exit SPOOL environment)
TIME <secs>     (To wait for phantom)
CANCEL <filename>
FINISH    (Stop when current file done)
LIST
```

External command.

START -- START EXECUTION

```
Start [<token>... ] [<p>] [<a>] [<b>] [<x>] [<k>]
```

Internal command.

STARTUP -- STARTUP DISK(S)

```
STARTUp [<nodename>] <dvno> ...
```

System user only.
Internal command.

STATUS -- PRINT USER OR SYSTEM STATUS

```
STATus [ ALl ]
       [USers]
       [DIsks]
       [UNits]
       [ NEt ]
```

Internal command.

SVCSW -- SET USER SVC SWITCH (INTERNAL

```
SVcsw [ 1 ]
      [ Ø ]
```

1 => bounce (except class 5), Ø => don't bounce.

Internal command.

TA -- TREE ATTACH

```
TA <treename>
```

WARNING: Wipes out memory.

External command.

TAPXAM -- READ/WRITE MAGTAPE

TAPXAM

Allows examination of 7 or 9-track tapes at a single tape command level. Can be used to diagnose failing drive or controller. Prompts for drive number, 7 or 9 track indication, maximum record size to read or write. Defaults (just hit .CR.) are drive 0, 9 track, '1000 words. Displays command, status, and NW transferred after each operation. Internal commands:

BF or BT	Backspace one filemark (tapemark).
BS or BR	Backspace one record.
DS	Display Status. Interprets last status read in readable form.
Eol	Write MAGSAV-style End-Logical-Tape record. Must first read last record on tape (to get sequence number right).
FF or FT	Forward space to file mark.
FS or FR	Forward space one record.
In	Re-initialize (all prompts repeated).
Max [<n>]	Set max amt of data to display after reads -- 0-10 words, defaults to 0.
Psd	Enter PSD (to examine block just read). Prints addr of buffer, use 'Q' in PSD to return to TAPXAM.
Quit	Exit TAPXAM (tape is not rewound).
R[C][l] [<nw>]	Read [Binary] [1 char/word] or [Correct] [1 char/word].
RW	Rewind to load point.
St	Read current status.
W[C][l] [<nw>]	Write [Binary] [1 char/word] or [Correct] [1 char/word].
WF or WT	Write file mark (tape mark).
1600BPI	Set density to 1600 bpi. (only applicable for software settable density)
6250BPI	Set density to 6250 bpi. (only applicable for software settable density)

External command.

TERM -- SET TERMINAL CHARACTERISTICS

```
TERM [ DISPLA ]
     [ ERASE  <char> ]
     [ KILL   <char> ]
     [ BREAK ON  ]
     [       OFF ]
     [ HALF [NOLF] [XOFF] ]
     [ FULL [XOFF] ]
```

External command.

TIME -- PRINT TIME STATISTICS

Time

HH'MM logged in, MM'SS CPU time, MM'SS I/O time.

Internal command.

TRAMLC -- AMLC I/O

```
TRAMLC TRANSMIT <filename> <line> [T]
TRAMLC RECEIVE <filename> <line> [T]
```

External command.

UNASSIGN -- RELEASE PERIPHERAL DEVICE

Unassign <parameters-identical-to-ASSIGN-command>

Internal command.

UPCASE -- TRANSLATE FILE TO UPPER CASE

```
UPCASE <intreename> [<outtreename>]
                    [<file-open-on-unit 2>]
```

External command.

USERS -- PRINT CURRENT NUMBER OF USERS

USErs

Internal command.

USRASR -- CONNECT SYSTEM ASR TO USER

```
USRasr <usrno>
```

Must type full USRASR if not logged in.

System user only.
Internal command.

VPSD -- VIRTUAL MODE PSD

```
VPSD
```

Supports V-mode. For internal commands, see PSD.

External command.

VRTSSW -- SET VIRTUAL SENSE SWITCHES

```
Vrtssw [<sense-switch-setting>]
       [          Ø           ]
```

Internal command.

4 FILE SYSTEM INTERNALS

The following describes the internal formats of all disk records for both the old and new file system partitions. All numbers are decimal unless preceded by a ':'. Where possible, field names are the same as those used by the internal file system routines.

DSKRAT FORMATS

DSKRAT Format -- Old Partitions

```
     ________
0  |        5 |  NUMBER OF WORDS IN DSKRAT HEADER = 5
1  |  RECSIZ  |  DISK RECORD SIZE (448 or 1040)
2  |  NMRECS  |  NUMBER RECORDS IN PARTITION
3  |  UNUSED  |  UNUSED
4  |  NHEADS  |  NUMBER HEADS IN PARTITION
5  |   DATA   |  START OF DKSRAT DATA (ONE BIT/RECORD)
   |   ....   |
```

DSKRAT Format -- New Partitions

```
     ________
0  |        8 |  NUMBER WORDS IN HEADER = 8
1  |  RECSIZ  |  RECORD SIZE
2  |  NMRECS  |  NUMBER RECORDS IN PARTITON (TWO WORDS)
   |__________|
4  |  NHEADS  |  NUMBER HEADS IN PARTITION
5  | RESERVED |  RESERVED
6  | RESERVED |  RESERVED
7  | RESERVED |  RESERVED
8  |   DATA   |  START OF DSKRAT DATA (ONE BIT/RECORD)
   |   ....   |
```

RECORD HEADER FORMATS

NOTE: record header formats are the same for new and old partitions. The format of a record header is a function of the physical record size.

Record Header Format -- 448-Word Records

```
     ________
0  |  REKCRA  |  RECORD ADDRESS (OF THIS RECORD)
1  |  REKBRA  |  RA OF DIRECTORY ENTRY OR FIRST RECORD
2  |  REKFPT  |  RA OF NEXT SEQUENTIAL RECORD
3  |  REKBPT  |  RA OF PREVIOUS RECORD
4  |  REKCNT  |  NUMBER DATA WORDS IN THIS RECORD
5  |  REKTYP  |  TYPE OF THIS FILE
6  |  REKLVL  |  INDEX LEVEL FOR NEW PART DAM FILES
7  | RESERVED |  RESERVED
```

Record Header Format -- 1040-Word Records

```
      ________
 0  |  REKCRA  |  RECORD ADDR OF THIS RECORD (TWO WORDS)
    |__________|
 2  |  REKBRA  |  BEGINNING RECORD ADDRESS (TWO WORDS)
    |__________|     (BRA OF DIRECTORY IF FIRST RECORD)
 4  |  REKCNT  |  NUMBER DATA WORDS IN THIS RECORD
 5  |  REKTYP  |  TYPE OF THIS FILE
 6  |  REKFPT  |  RA NEXT SEQUENTIAL RECORD (TWO WORDS)
    |__________|     (0 IF LAST)
 8  |  REKBPT  |  RA OF PREVIOUS RECORD (TWO WORDS)
    |__________|     (0 IF FIRST)
10  |  REKLVL  |  INDEX LEVEL FOR NEW PART DAM FILES
11  |          |
    |          |
    | RESERVED |  RESERVED (FIVE WORDS)
    |          |
15  |__________|
```

Notes

1) All disks except Storage Module have 448-word records. Storage Modules have 1040-word records.
2) The BRA of the first record in a file points to the beginning record address of the directory in which the file entry appears. In all other records, the BRA points to the first record of the file.
3) REKFPT contains the address of the next sequential record in the file or 0 if it is the last record in the file.
4) REKBPT contains the address of the previous record in sequence or 0 if it is the first record in the file.
5) REKTYP is valid only in the first record of a file. Legal values are:

0	SAM File
1	DAM File
2	SAM Segment Directory
3	DAM Segment Directory
4	User File Directory (UFD)

6) If the file is the record zero bootstrap (BOOT) or the disk record availability table (DSKRAT or volume name) and the disk has a 1040 record size (Storage Module), bit 1 (:100000) of FILTYP will be set.
7) DAM files on new partitions are organized somewhat differently from the above -- see PE-T-276.

UFD HEADER AND ENTRY FORMATS

Old UFD Header Format

```
    ________
Ø  |_______8|  SIZE OF HEADER -- 8 WORDS
1  | OPASSW |  OWNER PASSWORD (THREE WORDS)
   |        |
   |________|
4  | NPASSW |  NON-OWNER PASSWORD (THREE WORDS)
   |        |
   |________|
7  |RESERVED|  RESERVED
```

New UFD Header Format

```
    ________
Ø  |__ECW___|  ECW (SEE NOTE 1 ON NEXT PAGE)
1  | OPASSW |  OWNER PASSWORD (THREE WORDS)
   |        |
   |________|
4  | NPASSW |  NON-OWNER PASSWORD (THREE WORDS)
   |        |
   |________|
7  |        |
   |        |
   |RESERVED|  RESERVED (SIXTEEN WORDS)
   |        |
23 |________|
```

Old UFD Entry Format

```
    ________
Ø  |__BRA___|  BEGINNING RECORD ADDRESS
1  |  FILE  |  FILENAME (THREE WORDS)
   |        |
   |__NAME__|
4  |_SPACES_|  2 BLANKS FOR NAME EXPANSION (RESERVED)
5  |_PROTEC_|  PROTECTION (OWNER/NON-OWNER)
```

Notes

In an old UFD, the high-order eight bits of PROTEC are the owner rights stored in complemented form (Ø=>owner has right). The low-order eight bits are non-owner protection, stored in true form (Ø=>no right). On creation, PROTEC=Ø. After a 'PROT 7 Ø', PROTEC=:174ØØØ.

New UFD Entry Format

```
     ________
Ø   |__ECW___|  ENTRY CONTROL WORD (TYPE/LENGTH)
1   |  BRA   |  BEGINNING RECORD ADDRESS (TWO WORDS)
    |________|
3   |RESERVED|  RESERVED (THREE WORDS)
    |        |
    |________|
6   |_PROTEC_|  PROTECTION (OWNER/NON-OWNER)
7   |RESERVED|  RESERVED FOR FUTURE USE
8   |_DATMOD_|  DATE LAST MODIFIED (YYYYYYYMMMMDDDDD)
9   |_TIMMOD_|  TIME LAST MODIFIED (SECS-SINCE-MID./4)
1Ø  |_FILTYP_|  FILETYPE
11  |__SCW___|  SUBENTRY CONTROL WORD FOR FILENAME
12  |F       |
    |  I     |
    |    L   |
    |      E |
    |  ...   |  FILENAME (1-16 WORDS, BLANK PADDED)
    |N       |
    |  A     |
    |    M   |
N   |______E_|
```

Notes

1) The Entry Control Word (ECW) consists of two eight-bit subfields. The top eight bits indicate the type of the following entry as follows:

Ø	Old UFD Header
1	New UFD Header
2	Vacant Entry
3	New UFD File Entry

The low-order eight bits give the size of the entry including the ECW itself.

2) The bits in PROTEC are stored in true form (Ø=> no right) for both owner and non-owner fields.

3) The file type field is as before (see Old Record Header Format) with following additional bits:

BIT	MEANING WHEN BIT IS ON
1	File has 16-word hdr (DSKRAT, BOOT only).
4	Special file (BOOT, DSKRAT, MFD, BADSPT).

4) The Subentry Control Word (SCW) consists of two eight-bit subfields. The top 8 bits are Ø, indicating subentry type Ø. The low-order 8 bits give the size of the subentry including the SCW itself.

5) N.B.: UFD entries are reused by the new file system. This means that a new entry will not necessarily appear at the end of the UFD.

SEGMENT DIRECTORY FORMATS

Old Segment Directory Format

```
     _________
Ø  |  BRAØ   |  BRA OF FIRST ENTRY IN DIRECTORY
1  |  BRA1   |  BRA OF SECOND FILE
2  |  ØØØØ   |  NULL ENTRY
   |  ....   |
N  |  BRAn   |  BRA OF LAST FILE IN DIRECTORY
     ‾‾‾‾‾‾‾‾‾
```

New Segment Directory Format

```
     _________
Ø  |  BRAØ   |  BRA OF FIRST FILE IN DIR (2 WORDS)
   |_________|
2  |  BRA1   |  BRA OF SECOND FILE IN DIR (2 WORDS)
   |_________|
   |  ØØØØ   |  NULL ENTRY (2 WORDS)
   |  ØØØØ   |
   |  ....   |
   |_________|
2N |  BRAn   |  BRA OF LAST FILE IN DIR (2 WORDS)
   |_________|
```

Notes

The only difference between old and new directories is that each entry in a new directory is two words. A null entry in a new directory is a 32-bit Ø.

5 INSTRUCTION SET

The following description applies only to P4ØØ memory reference. The first number in the "opcode" column is the octal representation of instruction bits 3-6. The second number is the octal representation of bits 13-14 (bits 13-14 are inspected only if bits 6-11 are 11ØØØ, i.e., S=1 and -256 ≤ D < -224).

The 'type' column indicates the format and/or function of the operation as follows.

AP: Three-word operation, the last two words of which are an AP address pointer.
BR: Two-word operation, the second word of which is a word number within the current procedure segment to which to branch.
CON: Single-word control operation.
DA: Decimal arithmetic operation.
FE: Field and edit operation.
FLD: Single-word field operation.
FOP: Single-word floating-point operation.
FSK: Single-word floating-point skip operation.
IG: Single-word integrity operation.
IO: Single-word input/output operation.
LOG: Single-word logicize operation.
MOD: Single-word mode operation.
MR: Memory-reference operation.
OPR: Single-word miscellaneous operation.
PIO: Programmed input/output operation.
SH: Single-word shift operation.
SKP: Single-word skip operation.
VM: Virtual memory operation.

The 'C' column indicates the effect of the operation on the C-bit and the L-bit as follows.

-: C and L are unchanged by the operation.
1: C is unchanged, L is carry.
2: C is overflow, L is carry.
3: C is overflow, L is indeterminant.
4: C is shift extension, L is indeterminant.
5: C is a result of op, L is indeterminant.
6: C and L are indeterminant.
7: C and L are loaded by the operation.
8: C is cleared, L is indeterminant.

The 'cc' column indicates the effect of the operation on the condition codes as follows.

-: Cond. codes are unchanged.
1,4: Cond. codes result of arith op or compare.
5: Cond. codes indeterminant.
6: Cond. codes loaded by operation.
7: Cond. codes indicate result of operation.

The 'avail' column indicates in which addressing modes the operation is available as follows.

S: Available in 16S and 32S modes.
R: Available in 32R and 64R modes.
V: Available in 64V mode.
I: Available in 32I mode.
*: Restricted to Ring Ø execution.

PRIME 4ØØ MEMORY-REFERENCE INSTRUCTIONS
(WHEN IN 64V MODE).

	INSTRUCTION BITS 13-14 (if S=1 and -256<=D<-224)			
INSTRUCTION BITS 3-6	ØØ	Ø1	1Ø	11
ØØØ1	JMP	EAL	XEC	-
ØØ1Ø	LDA	FLD	DFLD	LDL
ØØ11	ANA	STLR	ORA	ANL
Ø1ØØ	STA	FST	DFST	STL
Ø1Ø1	ERA	LDLR	-	ERL
Ø11Ø	ADD	FAD	DFAD	ADL
Ø111	SUB	FSB	DFSB	SBL
1ØØØ	JST	-	PCL	-
1ØØ1	CAS	FCS	DFCS	CLS
1Ø1Ø	IRS	MIA	EAXB	-
1Ø11	IMA	MIB	EALB	-
11ØØ	JSY	EIO	JSXB	-
11Ø1*	STX	FLX	DFLX	-
11Ø1**	LDX	LDY	STY	JSX
111Ø	MPY	FMP	DFMP	MPL
1111	DIV	FDV	DFDV	DVL

Use column ØØ if S (bit 7) is Ø or if D (bits 8-16) is not in the range -256 <= D < -224.

*: Use this row if bit 2 of instr. word is a Ø. These instructions cannot be indexed.

**: Use this row if bit 2 of the instruction word is a 1. These instructions cannot be indexed.

				C		
MNEM	OPCODE	TYP	C	c	AVAIL	DESCRIPTION
A					I	ADD WORD.
A1A	141206	OPR	2	1	SRV	ADD 1 TO A REG. A+1=>A.
A2A	140304	OPR	2	1	SRV	ADD 2 TO A REGISTER. A+2=>A.
ABQ	50134	OPR	-	7	I	ADD TO BOTTOM OF QUEUE.
ABQ	141716	AP	-	6	SRV	ADD TO BOT OF Q. CCEQ => FULL.
ACA	141216	OPR	2	1	SRV	ADD CBIT TO A REG. CBIT+A=>A.
ADD	Ø6	MR	2	1	SRV	ADD. (A) + [EA]16 => A.
ADL	Ø6 Ø3	MR	2	1	V	(A,B)+[EA]32=>A,B. (NO HOLE).
ADLL	141ØØØ	OPR	2	1	SRV	ADD LINK TO L REGISTER.
ADLR	5ØØ14	OPR	-	7	I	ADD LINK TO GR.
AH	12	MR	2	1	I	ADD HALFWD. RH+[EA16]=>RH.
ALFA	ØØ13Ø1	FLD	6	5	SRV	ADD L TO FIELD ADDR REG. ZERO.
ALFA	ØØ1311	FLD	6	5	SRV	ADD L TO FIELD ADDR REG. ONE.
ALL	Ø414XX	SH	4	5	SRV	A LEFT LOGICAL.
ALR	Ø416XX	SH	4	5	SRV	A LEFT ROTATE.
ALS	Ø415XX	SH	4	5	SRV	A LEFT SHIFT (SHORT INT ARITH).
ANA	Ø3	MR	-	-	SRV	AND. (A) .AND. [EA]16 => A.
ANL	Ø3 Ø3	MR	-	-	V	(A,B) .AND. [EA]32 => A,B.
ARFA	5Ø161	FLD	-	7	I	ADD GR TO FIELD ADDR REG.
ARGT	ØØØ6Ø5	CON	-	-	SRV	ARG TRANSFER (USED WITH PCL).
ARL	Ø4Ø4XX	SH	4	5	SRV	A RIGHT LOGICAL.
ARR	Ø4Ø6XX	SH	4	5	SRV	A RIGHT ROTATE.
ARS	Ø4Ø5XX	SH	4	5	SRV	A RIGHT SHIFT (SHORT ARITH).
ATQ	5Ø135	OPR	-	7	I	ADD TO TOP OF QUEUE.
ATQ	141717	AP	-	6	SRV	ADD TO TOP OF Q. CCEQ => FULL.
BCEQ	1416Ø2	BR	-	-	SRV	BRANCH ON CONDITION CODE .EQ.
BCGE	1416Ø5	BR	-	-	SRV	BRANCH ON CONDITION CODE .GE.
BCGT	1416Ø1	BR	-	-	SRV	BRANCH ON CONDITION CODE .GT.
BCLE	1416ØØ	BR	-	-	SRV	BRANCH ON CONDITION CODE .LE.
BCLT	1417Ø4	BR	-	-	SRV	BRANCH ON CONDITION CODE .LT.
BCNE	1416Ø3	BR	-	-	SRV	BRANCH ON CONDITION CODE .NE.
BCR	1417Ø5	BR	-	-	SRV	BRANCH ON CBIT RESET.
BCS	1416Ø4	BR	-	-	SRV	BRANCH ON CBIT SET.
BDX	14Ø734	BR	-	-	SRV	BRANCH ON DECREMENTED X.
BDY	14Ø724	BR	-	-	SRV	BRANCH ON DECREMENTED Y.
BEQ	14Ø612	BR	-	4	SRV	BRANCH ON A REGISTER .EQ. Ø.
BFEQ	5Ø122	BR	-	4	I	BRANCH ON FLTG REG EQ.
BFEQ	141612	BR	-	4	SRV	BRANCH ON FAC .EQ. Ø.
BFGE	5Ø125	BR	-	4	I	BRANCH ON FLTG REG NE.
BFGE	141615	BR	-	4	SRV	BRANCH ON FAC .GE. Ø.
BFGT	5Ø121	BR	-	4	I	BRANCH ON FLTG REG LE.
BFGT	141611	BR	-	4	SRV	BRANCH ON FAC .GT. Ø.
BFLE	5Ø12Ø	BR	-	4	I	BRANCH ON FLTG REG LT.
BFLE	14161Ø	BR	-	4	SRV	BRANCH ON FAC .LE. Ø.
BFLT	5Ø124	BR	-	4	I	BRANCH on FLTG REG GT.
BFLT	141614	BR	-	4	SRV	BRANCH ON FAC .LT. Ø.
BFNE	5Ø123	BR	-	4	I	BRANCH ON FLTG REG GE.
BFNE	141613	BR	-	4	SRV	BRANCH ON FAC .NE. Ø.
BGE	14Ø615	BR	-	4	SRV	BRANCH ON A REGISTER .GE. Ø.
BGT	14Ø611	BR	-	4	SRV	BRANCH ON A REGISTER .GT. Ø.
BHD1	5Ø144	BR	-	-	I	BRANCH ON HALF REG DEC BY 1.
BHD2	5Ø145	BR	-	-	I	BRANCH ON HALF REG DEC BY 2.
BHD4	5Ø146	BR	-	-	I	BRANCH ON HALF REG DEC BY 4.
BHEQ	5Ø112	BR	-	4	I	BRANCH ON HALF REG EQ.
BHGT	5Ø111	BR	-	4	I	BRANCH ON HALF REG LE.
BHI1	5Ø14Ø	BR	-	-	I	BRANCH ON HALF REG INCR BY 1.
BHI2	5Ø141	BR	-	-	I	BRANCH ON HALF REG INCR BY 2.
BHI4	5Ø142	BR	-	-	I	BRANCH ON HALF REG INCR BY 4.
BHLE	5Ø11Ø	BR	-	4	I	BRANCH ON HALF REG LT.
BHLT	5Ø1Ø4	BR	-	4	I	BRANCH ON HALF REG GT.
BHNE	5Ø113	BR	-	4	I	BRANCH ON HALF REG GE.
BHNE	5Ø1Ø5	BR	-	4	I	BRANCH ON HALF REG NE.
BIX	141334	BR	-	-	SRV	BRANCH ON INCREMENTED X.
BIY	141324	BR	-	-	SRV	BRANCH ON INCREMENTED Y.
BLE	14Ø61Ø	BR	-	4	SRV	BRANCH ON A REGISTER .LE. Ø.
BLEQ	14Ø7Ø2	BR	-	4	SRV	BRANCH ON L REGISTER .EQ. Ø.
BLGE	14Ø615	BR	-	4	SRV	BRANCH ON L REGISTER .GE. Ø.
BLGT	14Ø7Ø1	BR	-	4	SRV	BRANCH ON L REGISTER .GT. Ø.
BLLE	14Ø7ØØ	BR	-	4	SRV	BRANCH ON L REGISTER .LE. Ø.
BLLT	14Ø614	BR	-	4	SRV	BRANCH ON L REGISTER .LT. Ø.
BLNE	14Ø7Ø3	BR	-	4	SRV	BRANCH ON L REGISTER .NE. Ø.
BLR	1417Ø7	BR	-	-	SRV	BRANCH ON LINK RESET.
BLS	1417Ø6	BR	-	-	SRV	BRANCH ON LINK SET.
BLT	14Ø614	BR	-	4	SRV	BRANCH ON A REGISTER .LT. Ø.
BMEQ	1416Ø2	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .EQ.
BMGE	1417Ø6	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .GE.
BMGT	14171Ø	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .GT.
BMLE	141711	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .LE.
BMLT	1417Ø7	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .LT.
BMNE	1416Ø3	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .NE.
BNE	14Ø613	BR	-	4	SRV	BRANCH ON A REGISTER .NE. Ø.
BRBR	5ØØ4Ø	BR	-	-	I	BRANCH ON REG BIT RESET.
BRBS	5ØØØØ	BR	-	-	I	BRANCH ON REG BIT SET.
BGR1	5Ø134	BR	-	-	I	BRANCH ON REG DEC BY 1.
BGR2	5Ø135	BR	-	-	I	BRANCH ON REG DEC BY 2.
BGR4	5Ø136	BR	-	-	I	BRANCH ON REG DEC BY 4.
BREQ	5Ø1Ø2	BR	-	4	I	BRANCH ON REG EQ.
BRGE	5Ø1Ø5	BR	-	4	I	BRANCH ON REG NE.
BRGT	5Ø1Ø1	BR	-	4	I	BRANCH ON REG LE.
BRI1	5Ø13Ø	BR	-	-	I	BRANCH ON REG INCR BY 1.
BRI2	5Ø131	BR	-	-	I	BRANCH ON REG INCR BY 2.
BRI4	5Ø132	BR	-	-	I	BRANCH ON REG INCR BY 4.
BRLE	5Ø1ØØ	BR	-	4	I	BRANCH ON REGISTER LT.
BRLT	5Ø1Ø4	BR	-	4	I	BRANCH ON REGISTER GT.
BRNE	5Ø1Ø3	BR	-	4	I	BRANCH ON REGISTER GE.
C	61	MR	1	1	I	COMPARE R WITH [EA32].
CAI	ØØØ411	IO	-	-	SRVI	*CLEAR ACTIVE INTERRUPT.
CAL	141Ø5Ø	OPR	-	-	SRV	CLEAR A REG. LEFT BYTE.
CALF	ØØØ7Ø5	AP	7	6	SRV	PROC CALL FROM FAULTING PROC.
CAR	141Ø44	OPR	-	-	SRV	CLEAR A REG. RIGHT BYTE.
CAS	11	MR	1	1	SRV	SKIP Ø,1,2 IF (A) >,=.< [EA]16.
CAZ	14Ø214	OPR	1	1	SRV	SKIP Ø,1,2 INST. IF A >,=,< Ø.
CEA	ØØØ111	OPR	-	-	SRV	A AS EA=>A. (USELESS IN 64V).
CGT	5ØØ26	OPR	-	7	I	COMPUTED GOTO.
CGT	ØØ1314	-	6	5	SRV	COMPUTED GO TO.
CH	71	MR	1	1	I	COMPARE RH WITH [EA16].
CHS	5ØØ4Ø	OPR	-	-	I	CHANGE SIGN -(1)=>GR(1).
CHS	14ØØ24	OPR	-	-	SRV	CHANGE SIGN OF A REGISTER.
CLS	11 Ø3	MR	1	1	V	SKIP Ø,1,2 IF (A,B)>,=.<[EA]32.

CMA	140401	OPR	-	-	SRV	ONE'S COMPLEMENT A REGISTER.
CMH	50045	OPR	-	-	I	COMP HALF REG. GRH=>GRH.
CMR	50044	OPR	-	-	I	COMP REG. GR=>GR.
CR	50056	OPR	-	-	I	CLEAR REG. 0 => GR.
CRA	140040	OPR	-	-	SRV	CLEAR A REGISTER. 0=>A.
CRB	140015	OPR	-	-	SRV	CLEAR B REGISTER. 0=>B.
CRBL	50062	OPR	-	-	I	LEFT BYTE 0=>GRH(1-8).
CRBR	50063	OPR	-	-	I	RIGHT BYTE 0=>GRH(9-16).
CRE	141404	OPR	-	-	SRV	CLEAR E. 0=>E.
CREP	10 02	MR	-	-	R	(P) => [(S)+1]16, EA=>P.
CRL	140010	OPR	-	-	SRV	CLEAR L REGISTER. 0=>L.
CRLE	141410	OPR	-	-	SRV	CLEAR L AND E. 0=>L, 0=>E.
CSA	140320	OPR	5	-	SRV	CPY SIGN OF A. A1=>CBIT,0=>A1.
CSR	50041	OPR	-	-	I	COPY & SAVE SIGN. 1=>C,0=>GR1.
CXCS	001714	IG	-	-	V	CONTROL EXTENDED CONTROL STORE.
D	62	MR	3	1	I	DIV. (R,R+1)/[EA32=>R RMR=>R+1.
DAD	06(DP)	MR	2	1	SR	(A,B)+[EA]32 => A,B.(W/HOLE).
DBL	000007	MOD	-	-	SRV	ENTER DOUBLE-PREC MODE.
DBLE	50106	FOP	-	-	I	CONV SINGLE TO DOUBLE FLTG PT.
DECH	70	MR	-	1	I	DECR HALFWD. [EA16]-1=>[EA16].
DFA	15 17	MR	3	1	I	DBLE FLTG ADD. DFR+[EA64]=>DFR.
DFAD	06 02	MR	3	5	RV	(DFAC) + [EA]64 => DFAC.
DFC	05 07	MR	-	1	I	DBLE FLTG COMP DRF TO [EA64].
DFCM	50144	FOP	3	1	I	DBL PRC FLTG COMP. -DFGR=>DFGR.
DFCM	140574	FOP	3	5	SRV	-DFAC=>DFAC.
DFCS	11 02	MR	6	5	RV	SKP 0,1,2 IF (DFAC)>,=.<[EA]64.
DFD	31,33	MR	3	1	I	DOUBLE FLTG DIVIDE.
DFDV	17 02	MR	3	5	RV	(DFAC) / [EA]64 => DFAC.
DFL	01 03	MR	-	-	I	DBLE FLTG LOAD. [EA643]=>DFR.
DFLD	02 02	MR	-	-	RV	[EA]64 => DFAC.
DFLX	15 02	MR	-	-	V	LD DFLT INDEX. 4*[EA]16 => X.
DFM	25 27	MR	3	1	I	DBL FLTG MULT. DFR/[EA64]=>DFR.
DFMP	16 02	MR	3	5	RV	(DFAC) * [EA]64 => DFAC.
DFS	21 23	MR	3	1	I	DBLE FLTG SUB. DFR-[EA64]=>DFR.
DFSB	07 02	MR	3	5	RV	(DFAC) - [EA]64 => DFAC.
DFST	11,13	MR	1	1	I	DBLE FLTG STORE. DFR=>[EA64].
DFST	04 02	MR	-	-	RV	(DFAC) => [EA]64.
DH	72	MR	3	1	I	DIV HW. R/[EA16]=>RH RM=>RL(2).
DH1	50130	OPR	2	1	I	DECR HALF REG BY 1. GRH-1=>GRH.
DH2	50131	OPR	2	1	I	DECR HALF REG BY 2. GRH-2=>GRH.
DIV	17	MR	3	5	V	(A,B)/[EA]16=>A,REM=>B.(NOHOLE)
DIV	17	MR	3	5	SR	(A,B)/[EA]16=>A;REM=>B.(W/HOLE)
DLD	02(DP)	MR	-	-	SR	DOUBLE LOAD. [EA]32 => A,B.
DM	60	MR	-	1	I	DECR. [EA32]-1=>[EA32].
DMH	70	MR	-	1	I	DECR HALDWD. [EA16]-1=>[EA16].
DR1	50124	OPR	2	1	I	DECR REG BY 1. GR-1=>GR.
DR2	50125	OPR	2	1	I	DECR REG BY 2. GR-2=>GR.
DRX	140210	OPR	-	-	SRV	DECREMENT X AND SKIP IF 0.
DSB	07(DP)	MR	2	1	SR	(A,B)-[EA]32 => A,B (W/HOLE).
DST	04(DP)	MR	-	-	SR	DOUBLE STORE. (A,B) => [EA]32.
DVL	17 03	MR	3	5	V	(A,B,E)/[EA]32=>A,B REM=>EH,EL.
E16S	000011	MOD	-	-	SRV	ENTER P300 16K SECTORED MODE.
E32I	001010	MOD	-	-	SRV	ENTER P500 321 MODE.
E32R	001013	MOD	-	-	SRVI	ENTER P300 32K RELATIVE MODE.
E32S	000013	MOD	-	-	SRVI	ENTER P300 32K SECTORED MODE.
E64R	001011	MOD	-	-	SRVI	ENTER P300 64K RELATIVE MODE.
E64V	000010	MOD	-	-	SRVI	ENTER P400 MODE.

EAA	01 01	MR	-	-	R	EFF. ADDR TO A-REG. EA => A.
EAFA	001300	AP	-	-	SRVI	EFF. ADDR TO FIELD REG 0.
EAFA	001310	AP	-	-	SRVI	EFF. ADDR TO FIELD REG 1.
EAL	01 01	MR	-	-	V	LOAD EFFECTIVE ADDR. EA => L.
EALB	42	MR	-	-	I	EFF ADDR to LB. EA=>LB.
EALB	13 02	MR	-	-	V	EFF. ADDR TO LB. EA => LB.
EAR	63	MR	-	-	I	EFF ADDR (EA32)TO R.
EAXB	52	MR	-	-	I	EFF ADDR TO XB. EA=>XB.
EAXB	12 02	MR	-	-	V	EFF. ADDR TO XB. EA => XB.
EIO	34	MR	-	2	I	EXECUTE ADDR TO BASE REG.
EIO	14 01	MR	-	7	V*	EXEC EA AS I/O INSTR.CCEQ=>SKP.
EMCM	000503	IG	-	-	SRVI*	ENTER MACH CHK MODE.
ENB	000401	IO	-	-	SRVI*	ENABLE INTERRUPTS.
ENTR	01 03	MR	-	-	R	(S)=>[(S)-EA]16, (S)-EA=>S.
EPMJ	000217	VM	-	-	SR	ENT PAGE MODE AND JUMP (P300).
EPMX	000237	VM	-	-	SR	ENT PAG MOD & JMP TO MICROCODE.
ERA	05	MR	-	-	SRV	(A) .XOR. [EA]16 => A.
ERL	05 03	MR	-	-	V	(A,B) .XOR. [EA]32 => A,B.
ERMJ	000701	VM	-	-	SR	ENTER RESTR MODE & JUMP (P300).
ERMX	000721	VM	-	-	SR	RESTR MOD & JUMP TO MICROCODE.
ESIM	000415	MOD	-	-	SRVI*	ENTER STANDAGR INTERRUPT MODE.
EVIM	000417	MOD	-	-	SRVI*	ENTER VECTORED INTERUPT MODE.
EVMJ	000703	VM	-	-	SR	ENT VIRT MODE AND JUMP (P300).
EVMX	000723	VM	-	-	SR	VIRT MOD & JUMP TO MICROCODE.
FA	14,16	MR	3	1	I	FLTG ADD. FR+[EA32]=>FR.
FAD	06 01	MR	3	5	RV	(FAC) + [EA]32 => FAC.
FC	04,06	MR	-	1	I	FLTG COMPARE FR TO [EA32]
FCM	50100	FOP	3	1	I	FLTG COMP. -FGR=>FGR.
FCM	140530	FOP	3	5	SRV	FLOATING COMP. -FAC=>FAC.
FCS	11 01	MR	6	5	RV	SKIP 0,1,2 IF (FAC)>,=.<[EA]32.
FD	30,32	MR	3	1	I	FLTG DIV. -FR/[EA32]=>FR.
FDBL	140016	FOP	-	-	SRV	FAC=>DFAC.
FDV	17 01	MR	3	5	RV	(FAC) / [EA]32 => FAC.
FLD	02 01	MR	-	-	RV	FLOATING LOAD. [EA]32 => FAC.
FLOT	140550	FOP	6	5	SRV	FLOT(A,B)=>FAC. (A,B) W/HOLE
FLT	50105	FOP	-	-	I	CONV INT TO FLTG PT FLOAT.
FLTA	140532	FOP	3	5	SRV	FLOT(A)=>FAC.
FLTH	50102	FOP	-	-	I	CNV HALF WD INT TO FLTG PT.
FLTL	140535	FOP	8	5	SRV	FLOT(L)=>FAC. (L W/NO HOLE)
FLX	15 01	MR	-	-	RV	2*[EA]16 => X.
FM	24,26	MR	3	1	I	FLTG MULT. FR*[EA32]=>FR.
FMP	16 01	MR	3	5	RV	(FAC) * [EA]32 => FAC.
FRN	50107	FOP	3	1	I	FLOATING ROUND UP.
FRN	140534	FOP	3	5	SRV	FLOATING ROUND UP.
FS	20,22	MR	3	1	I	FLTG SUB. FR-[EA32]=>FR.
FSB	07 01	MR	3	5	RV	(FAC) - [EA]32 => FAC.
FSGT	140515	FSK	-	4	SRV	FLOATING SKIP IF .GT. 0.
FSLE	140514	FSK	-	4	SRV	FLOATING SKIP IF .LE. 0.
FSMI	140512	FSK	-	4	SRV	FLOATING SKIP IF .LT. 0.
FSNZ	140511	FSK	-	4	SRV	FLOATING SKIP IF .NE. 0.
FSPL	140513	FSK	-	4	SRV	FLOATING SKIP IF .GE. 0.
FST	10,12	MR	-	-	I	FLTG STORE. FR=>[EA32].
FST	04 01	MR	6	6	RV	(FAC) => [EA]32.
FSZE	140510	FSK	-	4	SRV	FLOATING SKIP IF .EQ. 0.
HLT	000000	CON	-	-	SRVI*	HALT COMPUTER OPERATION.
I	41	MR	-	-	I	INTERCHANGE R WITH [EA32].
IAB	000201	OPR	-	-	SRV	EXCHANGE A AND B A=>B & B=>A.

ICA	14134Ø	OPR	-	-	SRV	INTERCHANGE BYTES OF A REG.
ICBL	5ØØ65	OPR	-	-	I	GRH(1-8)=>GRH(9-16), Ø=>R.
ICBR	5ØØ66	OPR	-	-	I	GRH(9-16)=>GRH(1-8), Ø=>
ICHL	5ØØ6Ø	OPR	-	-	I	GRH=>GRL, Ø=>GRH.
ICHR	5ØØ61	OPR	-	-	I	GRL=>GRH, Ø=>GRL.
ICL	14114Ø	OPR	-	-	SRV	EXCHANGE BYTES OF A, CLR LEFT.
ICR	14124Ø	OPR	-	-	SRV	EXCHANGE BYTES OF A, CLR RIGHT.
IH	51	MR	-	-	I	INTERCHANGE RH WITH [EA16].
IH1	5Ø126	OPR	2	1	I	GRH+1=>GRH.
IH2	5Ø127	OPR	2	1	I	GRH+2=>GRH.
ILE	141414	OPR	-	-	SRV	EXCHANGE L AND E. L=>E & E=>L.
IM	4Ø	I	-	1	I	INCR. [EA31]+1=>[EA32].
IMA	13	MR	-	-	SRV	EXCHANGE MEMORY AND A-REGISTER.
IMH	Ø	MR	-	1	I	INCR HALF WD. [EA16]+1=>[EA16].
INA	54	PIO	-	-	SR*	INPUT TO A-REGISTER.
INBC	ØØ1217	AP	6	5	SRVI*	NTFY IN INT, LIFO, DO CAI.
INBN	ØØ1215	AP	6	5	SRVI*	NTFY IN INT, LIFO, NO CAI.
INEC	ØØ1216	AP	6	5	SRVI*	NTFY IN INT, FIFO Q, DO CAI.
INEN	ØØ1214	AP	6	5	SRVI*	NTFY IN INT, FIFO Q, NO CAI.
INH	ØØ1ØØ1	IO	-	-	SRVI*	INHIBIT INTERRUPTS.
INK	5ØØ7Ø	OPR	-	-	I	MOVE KEYS TO REG KEYS.
INK	ØØØØ43	OPR	-	-	SRV	INPUT P-3ØØ KEYS INTO A REG.
INT	5Ø1Ø3	FOP	-	-	I	INT(FRS)=>GR.
INT	14Ø554	FOP	3	5	SRV	INT(FAC)=>A,B W/HOLE.
INTA	14Ø531	FOP	3	5	SRV	INT(FAC)=>A.
INTH	5Ø1Ø1	FOP	-	-	I	INT(FRS)=>GRH.
INTL	14Ø533	FOP	3	5	SRV	INT(FAC)=>L.
IR1	5Ø122	OPR	2	1	I	INCR REG BY 1. GR+1=>GR.
IR2	5Ø123	OPR	2	1	I	INCR REG BY 2. GR+2=>GR.
IRB	5ØØ62	OPR	-	-	I	BYTES GRH(1-8)=>GRH(9-16)
IRH	5ØØ57	OPR	-	-	I	GRH=>GRL, GRL=>GRH.
IRS	12	MR	-	-	SRV	INC, REPLACE, AND SKIP IF ZERO.
IRTC	ØØØ6Ø3	CON	7	6	SRVI*	INTERRUPT RETURN, DO CAI.
IRTN	ØØØ6Ø1	CON	7	6	SRVI*	INTERRUPT RETURN, NO CAI.
IRX	14Ø114	OPR	-	-	SRV	INCREMENT X AND SKIP IF Ø.
ITLB	ØØØ615	CON	-	-	SRVI*	INVAL STLB ENTRY, L = VADDR.
JDX	15 Ø2	MR	-	-	R	JUMP ON DECREMENTED X-R ZERO.
JEQ	Ø2 Ø3	MR	-	-	R	IF (A) .EQ. Ø, EA => P.
JGE	Ø7 Ø3	MR	-	-	R	IF (A) .GE. Ø, EA => P.
JGT	Ø5 Ø3	MR	-	-	R	IF (A) .GT. Ø, EA => P.
JIX	15 Ø3	MR	-	-	R	JUMP ON INCREMENTED X-REG ZERO.
JLE	Ø4 Ø3	MR	-	-	R	IF (A) .LE. Ø, EA => P.
JLT	Ø6 Ø3	MR	-	-	R	IF (A) .LT. Ø, EA => P.
JMP	51	MR	-	-	I	JUMP. EA=>RP.
JMP	Ø1	MR	-	-	SRV	UNCOND JUMP. EA => PB,P.
JNE	Ø3 Ø3	MR	-	-	R	IF (A) .NE. Ø, EA => P.
JSR	73	MR	-	-	I	RPL=>RH, EA32=>RP
JST	1Ø	MR	-	-	SRV	(P) => [EA]16, EA+1 => P.
JSX	35 Ø3	MR	-	-	RV	(P) => X, EA => P.
JSXB	61	MR	-	-	I	RP=>XB, EA=>RP.
JSXB	14 Ø2	MR	-	-	V	(PB,P) => XB, EA => PB,P.
JSY	14	MR	-	-	V	(P) => Y, EA => P.
L	Ø1	MR	-	-	I	LOAD. [EA32]=>R.
LCEQ	5Ø153	LOG	-	-	I	IF .EQ., 1=>GRH, ELSE Ø=>GRH.
LCEQ	1415Ø3	LOG	-	-	SRV	IF CC.EQ.,1=>A. ELSE Ø=>A.
LCGE	5Ø154	LOG	-	-	I	IF .GE.,1=>GRH, ELSEØ=>GRH.
LCGE	1415Ø4	LOG	-	-	SRV	IF CC.GE.,1=>A. ELSE Ø=>A.

LCGT	5Ø155	LOG	-	-	I	IF .GT., 1=>GRH, ELSE Ø=>GRH.
LCGT	1415Ø5	LOG	-	-	SRV	IF CC.GT.,1=>A. ELSE Ø=>A.
LCLE	5Ø151	LOG	-	-	I	IF .LE., 1=>GRH, ELSE Ø=>GRH.
LCLE	1415Ø1	LOG	-	-	SRV	IF CC.LE.,1=>A. ELSE Ø=>A.
LCLT	5Ø15Ø	LOG	-	-	I	IF .LT., 1=>GRH, ELSE Ø=>GRH.
LCLT	1415ØØ	LOG	-	-	SRV	IF CC.LT.,1=>A. ELSE Ø=>A.
LCNE	5Ø152	LOG	-	-	I	IF .NE.,1=>GRH, ELSE Ø=>GRH.
LCNE	1415Ø2	LOG	-	-	SRV	IF CC.NE.,1=>A. ELSE Ø=>A.
LDA	Ø2	MR	-	-	SRV	LOAD A-REGISTER. [EA]16 => A.
LDAR	44	MR	-	-	I	LOAD ADDR REGISTER.
LDC	5Ø162	FLD	-	7	I	LOAD CHAR TO GRH.
LDC	ØØ1312	FLD	-	7	SRV	LOAD CHAR TO A REG PER FAR 1.
LDC	ØØ13Ø2	FLD	-	7	SRV	LOAD CHAR TO A REG VIA FAR Ø.
LDL	Ø2 Ø3	MR	-	-	V	LOAD LONG. [EA]32 => A,B.
LDLR	Ø5 Ø1	MR	-	-	V	LOAD LONG FROM RFILE LOC EA.
LDX	35	MR	-	-	SRV	LOAD X-REGISTER. [EA]16 => X.
LDY	35 Ø1	MR	-	-	V	LOAD Y-REGISTER. [EA]16 => Y.
LEQ	5ØØØ3	LOG	-	4	I	IF GR=Ø TN 1=>GRH,ELSE Ø=>GRH.
LEQ	14Ø413	LOG	-	4	SRV	IF A.EQ.Ø, 1=>A. ELSE Ø=>A.
LF	5ØØ16	LOG	-	4	I	LOGICIZE .F. Ø=>GRH.
LF	14Ø416	LOG	-	5	SRV	LOGICIZE FALSE. Ø=>A.
LFEQ	5ØØ23	LOG	-	4	I	IF FRS=Ø TN 1=>GRH,ELSE Ø=>GRH.
LFEQ	141113	LOG	-	4	SRV	IF FAC.EQ.Ø, 1=>A. ELSE Ø=>A.
LFGE	5ØØ24	LOG	-	4	I	IF FRS>=Ø T 1=>GRH,ELSE Ø=>GRH.
LFGE	141114	LOG	-	4	SRV	IF FAC.GE.Ø, 1=>A. ELSE Ø=>A.
LFGT	5ØØ25	LOG	-	4	I	IF FRS>Ø TN 1=>GRH,ELSE Ø=>GRH.
LFGT	141115	LOG	-	4	SRV	IF FAC.GT.Ø, 1=>A. ELSE Ø=>A.
LFLE	5ØØ21	LOG	-	4	I	IF FRS<=Ø T 1=>GRH,ELSE Ø=>GRH.
LFLE	141111	LOG	-	4	SRV	IF FAC.LE.Ø, 1=>A. ELSE Ø=>A.
LFLI	ØØ1313	-	-	-	SRV	LOAD FIELD LEN REG IMM ONE.
LFLI	ØØ13Ø3	-	-	-	SRV	LOAD FIELD LEN REG IMM ZERO.
LFLT	5ØØ2Ø	LOG	-	4	I	IF FRS<Ø T 1=>GRH,ELSE Ø=>GRH.
LFLT	14111Ø	LOG	-	4	SRV	IF FAC.LT.Ø, 1=>A. ELSE Ø=>A.
LFNE	5ØØ22	LOG	-	4	I	IF FRS<>Ø T 1=>GRH,ELSE Ø=>GRH.
LFNE	141112	LOG	-	4	SRV	IF FAC.NE.Ø, 1=>A. ELSE Ø=>A.
LGE	5ØØØ4	LOG	-	4	I	IF GR>=Ø TN 1=>GRH,ELSE Ø=>GRH.
LGE	14Ø414	LOG	-	4	SRV	IF A.GE.Ø, 1=>A. ELSE Ø=>A.
LGT	5ØØØ5	LOG	-	4	I	IF GR>Ø TN 1=>GRH, ELSE Ø=>GRH.
LGT	14Ø415	LOG	-	4	SRV	IF A.GT.Ø, 1=>A. ELSE Ø=>A.
LH	11	MR	-	-	I	LOAD HALF WD. [EA16]=>RH.
LHEQ	5ØØ13	LOG	-	4	I	IF GRH=Ø TN 1=>GRH,ELSE Ø=>GRH.
LHGE	5ØØØ4	LOG	-	4	I	IF GRH>=Ø T 1=>GRH,ELSE Ø=>GRH.
LHGT	5ØØ15	LOG	-	4	I	IF GRH>Ø TN 1=>GRH,ELSE Ø=>GRH.
LHL1	Ø4	MR	-	-	I	SHIFTED 1 [EA16] .LS. 1=>GRH.
LHL2	14	MR	-	-	I	SHIFTED 2 [EA16] .LS. 2=>GRH.
LHLE	5ØØ11	LOG	-	4	I	IF GRH<=Ø T 1=>GRH,ELSE Ø=>GRH.
LHLT	5ØØØØ	LOG	-	4	I	IF GR<Ø TN 1=>GRH,ELSE Ø=>GRH.
LHNE	5ØØ12	LOG	-	4	I	IF GRH<>Ø T 1=>GRH,ELSE Ø=>GRH.
LLE	5ØØØ1	LOG	-	4	I	IF GR<=Ø TN 1=>GRH,ELSE Ø=>GRH.
LLE	14Ø411	LOG	-	4	SRV	IF A.LE.Ø, 1=>A. ELSE Ø=>A.
LLEQ	141513	LOG	-	4	SRV	IF L.EQ.Ø, 1=>A. ELSE Ø=>A.
LLGE	14Ø414	LOG	-	4	SRV	IF L.GE.Ø, 1=>A. ELSE Ø=>A.
LLGT	141515	LOG	-	4	SRV	IF L.GT.Ø, 1=>A. ELSE Ø=>A.
LLL	Ø41ØXX	SH	4	5	SRV	LONG LEFT LOGICAL.
LLLE	141511	LOG	-	4	SRV	IF L.LE.Ø, 1=>A. ELSE Ø=>A.
LLLT	14Ø41Ø	LOG	-	4	SRV	IF L.LT.Ø, 1=>A. ELSE Ø=>A.
LLNE	141512	LOG	-	4	SRV	IF L.NE.Ø, 1=>A. ELSE Ø=>A.

```
LLR  Ø412XX SH   4 5 SRV  LONG LEFT ROTATE.
LLS  Ø411XX SH   4 5 SRV  LNG LSFT 64V=>LNG INT,ELSE HOLE
LLT   5ØØØØ LOG - 4 I     IF GR<Ø TN 1=> GRH,ELSE Ø=>GRH.
LLT  14Ø41Ø LOG - 4 SRV   IF A.LT.Ø, 1=>A. ELSE Ø=>A.
LMCM ØØØ5Ø1 IG  - - SRVI*LEAVE MACHINE CHECK MODE.
LNE   5ØØØ2 LOG - 4 I     IF GR<>Ø TN 1=>GRH,ELSE Ø=>GRH.
LNE  14Ø412 LOG - 4 SRV   IF A.NE.Ø, 1=>A. ELSE Ø=>A.
LPID ØØØ617 CON - - SRVI*LOAD PROCESS ID FROM AØ1-A12.
LPMJ ØØØ215 VM  - - SR    LEAVE PAGE MODE & JUMP (P3ØØ).
LPMX ØØØ235 VM  - - SR    LVE PAG MOD & JMP TO MICROCODE.
LPSW ØØØ711 AP  7 6 SRVI*LOAD PSW (SN,WN,KEYS,MODALS).
LRL  Ø4ØØXX SH  4 5 SRV   LONG RIGHT LOGICAL.
LRR  Ø4Ø2XX SH  4 5 SRV   LONG RIGHT ROTATE.
LRS  Ø4Ø1XX SH  4 5 SRV   LNG RSFT.64V=LNG INT,ELSE HOLE.
LT   14Ø417 LOG - 5 SRV   LOGICIZE TRUE.  1=>A.
LWCS ØØ171Ø IG  - - V     LOAD WRITABLE CONTROL STORE.
M    42     MR  3 1 I     MULT. R*[EA32]=>(R,R+1).
MDEI ØØ13Ø4 IG  - - SRVI*MEM DIAG ENABLE INTERLEAVE.
MDII ØØ13Ø5 IG  - - SRVI*MEM DIAGN INHIBIT INTERLEAVE.
MDIW 0013?? IG  - - SRVI*MEM DIAG WRT INTERLV.  L=>[E].
MDRS ØØ13Ø6 IG  - - SRVI*MEM DIAG READ SYNDROME BITS.
MDWC ØØ13Ø7 IG  - - SRV* MEM DIAG LOAD WRITE CTL REG.
MH   52     MR  3 1 I     RH*[EA16}=>R.
MIA  64     MR  - - I     MICROCODE EXECUTE A.
MIA  12 Ø1  MR  - - V     MICROCODE ENTRANCE.
MIB  74     MR  - - I     MICROCODE EXECUTE B.
MIB  13 Ø1  MR  - - V     MICROCODE ENTRANCE.
MPL  16 Ø3  MR  - 1 V      (A,B) * [EA]16 => A,B.(NOHOLE)
MPY  16     MR  - 1 V      (A) * [EA]16 => A,B. (NOHOLE).
MPY  16     MR  3 1 SR     (A) * [EA]16 => A,B. (HOLE).
N    Ø3     MR  - - I     R .AND. [EA32]=>R.
NFYB ØØ1211 AP  6 5 SRVI*NOTIFY ON SEM AT AP.  LIFO Q.
NFYE ØØ121Ø AP  6 5 SRVI*NOTIFY ON SEM AT AP.  FIFO Q.
NH   13     MR  - - I     RH .AND. [EA16]=>RH.
NOP  ØØØØØ1 OPR - - SRV   NO OPERATION.
NRM  ØØØ1Ø1 OPR - - SRV   NORMALIZE A,B AS ON P-3ØØ.
O    23     MR  - - I     R .OR. [EA32]=>R.
OCP  14     PIO - - SR*   OUTPUT CONTROL PULSE.
OH   33     MR  - - I     RH .OR. [EA16]=>RH.
ORA  Ø3 Ø2  MR  - - V      (A) .OR. [EA]16 => A.
OTA  74     PIO - - SR*   OUTPUT FROM A-REGISTER.
OTK   5ØØ71 OPR 7 S I     SET KEYS FROM REGISTER
OTK  ØØØ4Ø5 OPR 7 6 SRV   OUTPUT A TO 3ØØ KEYS, SHFT CTR.
PCL  41     MR  - - I     PROCEDURE CALL.
PCL  1Ø Ø2  MR  7 6 V     P-4ØØ PROCEDURE CALL.
PID   5ØØ52 OPR - - I     GR=>GRN, (1)=>GR(2-32).
PID  ØØØ211 OPR - - SRV   SHORT INT TO DP INT W/HOLE.
PIDA ØØØ115 OPR - - SRV   SHORT TO LONG INT CONV. A=>L.
PIDH  5ØØ53 OPR - - I     GRH=>GRL,GRH(1)=>GRH(2-16).
PIDL ØØØ3Ø5 OPR - - SRV   CONVERT LONG INT TO 64 BIT INT.
PIM   5ØØ5Ø OPR 2 1 I     POS REG AFTER INT MULTIPLY.
PIM  ØØØ2Ø5 OPR - - SRV   DP INT WITH HOLE TO SHORT INT.
PIMA ØØØØ15 OPR 3 5 SRV   L=>A. IEX ON PREC LOSS.
PIMH  5ØØ51 OPR 2 1 I     POS HALF REG AFTER INT MULT.
PIML ØØØ3Ø1 OPR 3 5 SRV   64BIT INT TO LONG INT.(L,E)=>L.
PRTN ØØØ611 CON 7 6 SRVI  PROCEDURE RETURN.
RBQ   5Ø133 OPR - 7 I     REMOVE FROM BOTTOM OF QUEUE.
```

```
RBQ  141715 AP  - 6 SRV   RMV BOT OF Q. EMP=>A=Ø, CCEQ
RCB  14Ø2ØØ OPR 5 - SRV   RESET CBIT.  Ø=>CBIT.
RMC  ØØØØ21 IG  - - SRVI*RESET MACHINE CHECK FLAG.
RRST ØØØ717 AP  - - SRV   REST REGS (GEN, FLT, XB).
RSAV ØØØ715 AP  - - SRV   SAVE REGS (GEN, FLT, XB).
RTN  ØØØ1Ø5 OPR - - SRV   RETURN FROM P-3ØØ RECUR PROC.
RTQ   5Ø132 OPR - 7 I     REMOVE FROM TOP OF QUEUE.
RTQ  141714 AP  - 6 SRV   RMV TOP OF Q. EMP=>A=Ø, CCEQ.
S    22     MR  2 1 I     SUB. R-[EA32]=>R.
S1A  14Ø11Ø OPR 2 1 SRV   SUB 1 FROM A REGISTER.  A-1=>A.
S2A  14Ø31Ø OPR 2 1 SRV   SUB 2 FROM A REGISTER.  A-2=>A.
SAR  1ØØ26X SKP - - SRV   SKIP IF A REG. BIT N RESET.
SAS  1Ø126Ø SKP - - SRV   SKIP IF A REG. BIT N SET.
SBL  Ø7 Ø3  MR  2 1 V     (A,B) - [EA]32 => A,B.(NOHOLE)
SCA  ØØØØ41 OPR - - SRV   LOAD P-3ØØ SHFT CTR INTO A REG.
SCB  14Ø6ØØ OPR 5 - SRV   SET CBIT.  1=>CBIT.
SGL  ØØØØØ5 MOD - - SRV   ENTER SINGLE-PRECISION MODE.
SGT  1ØØ22Ø SKP - - SRV   SKIP IF A REG. .GT. Ø.
SH   32     MR  2 1 I     SUB HALF WD. RH-[EA16]=>RH.
SHA  15     MR  4 - I     ARITHMETIC SHIFT.
SHL  Ø5     MR  4 - I     LOGICAL SHIFT.
SHL1  5ØØ76 OPR 4 1 I     GRH .LS. 1 => GRH.
SHL2  5ØØ77 OPR 4 1 I     GRH .LS. 2 => GRH.
SHR1  5Ø12Ø OPR 4 1 I     GRH .RS. 1 => GRH.
SHR2  5Ø121 OPR 4 1 I     GRH .RS. 2 => GRH.
SKP  1ØØØØØ SKP - - SRV   SKIP ONE WORD.
SKS  34     PIO - - SR    SKIP IF CONDITION SET.
SL1   5ØØ72 OPR 4 1 I     GR .LS. 1 => GR.
SL2   5ØØ73 OPR 4 1 I     GR .LS. 2 => GR.
SLE  1Ø122Ø SKP - - SRV   SKIP IF A REG. .LE. Ø.
SLN  1Ø11ØØ SKP - - SRV   SKIP IF A REG. BIT 16 SET.
SLZ  1ØØ1ØØ SKP - - SRV   SKIP IF A REG. BIT 16 .EQ. Ø.
SMCR 1ØØ2ØØ SKP - - SRV   SKIP IF MACHINE CHECK RESET.
SMCS 1Ø12ØØ SKP - - SRV   SKIP IF MACHINE CHECK SET.
SMI  1Ø14ØØ SKP - - SRV   SKIP IF A REG. .LT. Ø.
SMK  17ØØ2Ø PIO - - SR*   SET INTERRUPT MASKS.
SNR  1ØØ24X SKP - - SRV* SKIP IF SENSE SWITCH N RESET.
SNS  1Ø124Ø SKP - - SRV* SKIP IF SENSE SWITCH N SET.
SNZ  1Ø1Ø4Ø SKP - - SRV   SKIP IF A REG. NE. Ø.
SPL  1ØØ4ØØ SKP - - SRV   SKIP IF A REG. .GE. Ø.
SR1   5ØØ74 OPR 4 1 I     GR .RS. 1 => GR.
SR1  1ØØØ2Ø SKP - - SRV* SKIP IF SENSE SWITCH 1 RESET.
SR2   5ØØ75 OPR 4 1 I     GR .RS. 2 => GR.
SR2  1ØØØ1Ø SKP - - SRV* SKIP IF SENSE SWITCH 2 RESET.
SR3  1ØØØØ4 SKP - - SRV* SKIP IF SENSE SWITCH 3 RESET.
SR4  1ØØØØ2 SKP - - SRV* SKIP IF SENSE SWITCH 4 RESET.
SRC  1ØØØØ1 SKP - - SRV   SKIP IF CBIT RESET.
SS1  1Ø1Ø2Ø SKP - - SRV* SKIP IF SENSE SWITCH 1 SET.
SS2  1Ø1Ø1Ø SKP - - SRV* SKIP IF SENSE SWITCH 2 SET.
SS3  1Ø1ØØ4 SKP - - SRV* SKIP IF SENSE SWITCH 3 SET.
SS4  1Ø1ØØ2 SKP - - SRV* SKIP IF SENSE SWITCH 4 SET.
SSC  1Ø1ØØ1 SKP - - SRV   SKIP IF CBIT SET.
SSM   5ØØ42 OPR - - I     SET SIGN. MINUS 1=>(1).
SSM  14Ø5ØØ OPR - - SRV   SET SIGN OF A MINUS. 1=>A1.
SSP   5ØØ43 OPR - - I     SET SIGN. PLUS Ø=>(1).
SSP  14Ø1ØØ OPR - - SRV   SET SIGN OF A PLUS.  Ø=>ABIT1.
SSR  1ØØØ36 SKP - - SRV* SKIP IF SSWI 1,2,3 AND 4 RESET.
```

SSS	1Ø1Ø36	SKP	-	-	SRV*	SKIP IF SSWI 1,2,3 AND 4 SET.
ST	21	MR	-	-	I	STORE. R=>[EA32].
STA	Ø4	MR	-	-	SRV	STORE A-REG. (A) => [EA]16.
STAC	ØØ12ØØ	AP	-	7	SRV	ST A COND ON B=[EA16].CCEQ= OK.
STAR	54	MR	-	-	I	STORE ADDRESSED REGISTER.
STC	5Ø166	FLD	-	7	I	STORE CHARACTER FROM GRH.
STC	ØØ1332	FLD	-	7	SRV	STORE CHAR FROM A (SEE FAR1).
STC	ØØ1322	FLD	-	7	SRV	STORE CHAR FROM A (SEE FARØ).
STCD	5Ø137	OPR	-	7	I	GR+1=[EA16].
STCH	5Ø136	OPR	-	7	I	GRL=[EA16].
STEX	5ØØ27	OPR	-	7	I	STACK EXTEND.
STEX	ØØ1315	OPR	6	5	SRV	STK EXTEND. L REG HAS EXTENT.
STFA	ØØ133Ø	AP	-	-	SRVI	STORE FIELD ADDR REG 1.
STFA	ØØ132Ø	AP	-	-	SRVI	STORE FIELD ADDR REG Ø.
STH	31	MR	-	-	I	STORE HALF WD. RH=>[EA16].
STL	Ø4 Ø3	MR	-	-	V	STORE LONG. (A,B) => [EA]32.
STLC	ØØ12Ø4	AP	-	7	SRV	ST L COND ON E=[EA32].CCEQ=OK.
STLR	Ø3 Ø1	MR	-	-	V	STORE LONG INTO RFILE LOC EA.
STX	15	MR	-	-	SRV	STORE X-REG. (X) => [EA]16.
STY	35 Ø2	MR	-	-	V	STORE Y-REG. (Y) => [EA]16.
SUB	Ø7	MR	2	1	SRV	SUBTRACT. (A) - [EA]16 => A.
SVC	ØØØ5Ø5	CON	-	-	SRVI	SUPERVISOR CALL.
SZE	1ØØØ4Ø	SKP	-	-	SRV	SKIP IF A REG .EQ. Ø.
TAB	14Ø314	OPR	-	-	SRV	TRANSFER A TO B REG. A=>B.
TAK	ØØ1Ø15	OPR	7	6	SRV	TRANSFER A TO KEYS.
TAX	14Ø5Ø4	OPR	-	-	SRV	TRANS A REG TO X REG. A=>X.
TAY	14Ø5Ø5	OPR	-	-	SRV	TRANS A REG TO Y REG. A=>Y.
TBA	14Ø6Ø4	OPR	-	-	SRV	TRANS B REG TO A REG. B=>A.
TC	5ØØ47	OPR	3	1	I	-GR+1=>GR.
TCA	14Ø4Ø7	OPR	2	1	SRV	TWO'S COMPLEMENT A. -A=>A.
TCH	5ØØ47	OPR	3	1	I	-GRH+1=>GRH.
TCL	14121Ø	OPR	2	1	SRV	TWO'S COMPLEMENT L. -L=>L.
TFLL	ØØ1333	FLD	-	-	SRV	XFER FLD LEN REG TO L REG 1.
TFLL	ØØ1323	FLD	-	-	SRV	XFER FLD LEN REG TO L REG Ø.
TFLR	5Ø163	FLD	-	7	I	TRANSFER FLD LENGTH REG TO GR.
TKA	ØØ1ØØ5	OPR	-	-	SRV	TRANSFER KEYS TO A.
TLFL	ØØ1321	FLD	-	-	SRV	TRANS L TO FLD LEN REG Ø.
TLFL	ØØ1331	FLD	-	-	SRV	TRANSFER L TO FLD LEN REG 1.
TM	44	MR	-	1	I	REST MEM. [EA32]::Ø=>CC.
TRFL	5Ø165	FLD	-	7	I	TRANSFER GR TO FLD LENGTH REG.
TSTQ	5Ø1Ø4	OPR	-	7	I	TEST QUEUE.
TSTQ	141757	AP	-	6	SRV	TEST Q. A=# ITEMS. CCEQ=> EMP.
TXA	141Ø34	OPR	-	-	SRV	TRANS X REG TO A REG. X=>A.
TYA	141124	OPR	-	-	SRV	TRANS Y REG TO A REG. Y=>A.
VIRY	ØØØ311	IG	5	6	SRV*	EXECUTE VERIFICATON ROUTINE.
WAIT	ØØØ315	AP	-	-	SRV*	WAIT ON SEMAPHORE AT AP.
WCS	ØØ16XX	-	-	-	SRV*	WCS ENTRANCES. UII ON NO WCS.
X	43	MR	-	-	I	EXC OR. R .XOR. [EA32]=>R.
XAD	ØØ11ØØ	DA	X	X	VI	ADD TWO DECIMAL FLDS.
XBTD	ØØ1145	DA	X	X	VI	CONV BI DEC VALUE TO DEC FLD.
XCA	14Ø1Ø4	OPR	-	-	SRV	EXCHG AND CLR A. A=>B, Ø=>A.
XCB	14Ø2Ø4	OPR	-	-	SRV	EXCHG AND CLR B. B=>A, Ø=>B.
XEC	Ø1 Ø2	MR	-	-	RV	EXECUTE INSTRUCTION AT EA.
XCM	ØØ11Ø2	DA	X	X	VI	COMP TWO NUMERIC FLDS.
XDTB	ØØ1146	DA	X	X	VI	CONV DEC FLD TO BI REG VALUE.
XDZ	ØØ11Ø7	DA	X	X	VI	DIV DEST FLD BY SOURCE FLD.
XEC	Ø1 Ø2	MR	-	-	RV	EX EFF ADDR CONT AS THIS INST.

XED	ØØ1112	FE	X	X	VI	EDIT NUMERIC FIELD.
XH	52	MR	-	-	I	RH .XOR. [EA16]=>RH.
XMP	ØØ11Ø4	DA	X	X	VI	MULTIPLY TWO DEC FLDS.
XMV	ØØ11Ø1	DA	X	X	VI	MOVE NUM SOURCE FLD TO DST FLD.
ZCM	ØØ1117	CS	X	X	VI	COMP TWO CHAR STR FIELDS.
ZED	ØØ1111	FE	X	X	VI	EDIT CHAR STR FIELD.
ZFIL	Ø11116	CS	X	X	VI	FILL CHAR STR FLD WITH CHAR.
ZM	43	MR	-	-	I	ZERO MEM. Ø=>[EA32].
ZMH	53	MR	-	-	I	ZERO MEM. HALF WD. Ø=>[EA16].
ZMV	ØØ1114	CS	X	X	VI	CPY FRM SOURCE FLD TO DEST FLD.
ZMVD	ØØ1115	-	X	X	VI	CPY FROM SOURCE FLD TO DEST FLD OF EQUAL LENGTH.
ZTRN	ØØ111Ø	CS	X	X	VI	TRANS SRCE STR FLD TO DEST FLD.
xxx	14ØØ14	OPR	-	-	SRV	OBSOLETE. CLRS B, LSW OF DFAC.

INSTRUCTION SET GROUPED BY FUNCTION

Miscellaneous Operations

CGT	ØØ1314	-	6	5	SRV	COMPUTED GO TO.
LFLI	ØØ1313	-	-	-	SRV	LOAD FIELD LEN REG IMM ONE.
LFLI	ØØ13Ø3	-	-	-	SRV	LOAD FIELD LEN REG IMM ZERO.
WCS	ØØ16XX	-	-	-	SRV*	WCS ENTRANCES. UII ON NO WCS.
ZMVD	ØØ1115	-	X	X	VI	CPY FROM SOURCE FLD TO DEST

Address Pointer Operations

ABQ	141716	AP	-	6	SRV	ADD TO BOT OF Q. CCEQ => FULL.
ATQ	141717	AP	-	6	SRV	ADD TO TOP OF Q. CCEQ => FULL.
CALF	ØØØ7Ø5	AP	7	6	SRV	PROC CALL FROM FAULTING PROC.
EAFA	ØØ13ØØ	AP	-	-	SRVI	EFF. ADDR TO FIELD REG Ø.
EAFA	ØØ131Ø	AP	-	-	SRVI	EFF. ADDR TO FIELD REG 1.
INBC	ØØ1217	AP	6	5	SRVI*	NTFY IN INT, LIFO, DO CAI.
INBN	ØØ1215	AP	6	5	SRVI*	NTFY IN INT, LIFO, NO CAI.
INEC	ØØ1216	AP	6	5	SRVI*	NTFY IN INT, FIFO Q, DO CAI.
INEN	ØØ1214	AP	6	5	SRVI*	NTFY IN INT, FIFO Q, NO CAI.
LPSW	ØØØ711	AP	7	6	SRVI*	LOAD PSW (SN,WN,KEYS,MODALS).
NFYB	ØØ1211	AP	6	5	SRVI*	NOTIFY ON SEM AT AP. LIFO Q.
NFYE	ØØ121Ø	AP	6	5	SRVI*	NOTIFY ON SEM AT AP. FIFO Q.
RBQ	141715	AP	-	6	SRV	RMV BOT OF Q. EMP=>A=Ø, CCEQ
RRST	ØØØ717	AP	-	-	SRV	REST REGS (GEN, FLT, XB).
RSAV	ØØØ715	AP	-	-	SRV	SAVE REGS (GEN, FLT, XB).
RTQ	141714	AP	-	6	SRV	RMV TOP OF Q. EMP=>A=Ø, CCEQ.
STAC	ØØ12ØØ	AP	-	7	SRV	ST A COND ON B=[EA16].CCEQ= OK.
STFA	ØØ133Ø	AP	-	-	SRVI	STORE FIELD ADDR REG 1.
STFA	ØØ132Ø	AP	-	-	SRVI	STORE FIELD ADDR REG Ø.
STLC	ØØ12Ø4	AP	-	7	SRV	ST L COND ON E=[EA32].CCEQ=OK.
TSTQ	141757	AP	-	6	SRV	TEST Q. A=# ITEMS. CCEQ=> EMP.
WAIT	ØØØ315	AP	-	-	SRV*	WAIT ON SEMAPHORE AT AP.

Branch Operations

BCEQ	1416Ø2	BR	-	-	SRV	BRANCH ON CONDITION CODE .EQ.
BCGE	1416Ø5	BR	-	-	SRV	BRANCH ON CONDITION CODE .GE.

BCGT	141601	BR	-	-	SRV	BRANCH ON CONDITION CODE .GT.
BCLE	141600	BR	-	-	SRV	BRANCH ON CONDITION CODE .LE.
BCLT	141704	BR	-	-	SRV	BRANCH ON CONDITION CODE .LT.
BCNE	141603	BR	-	-	SRV	BRANCH ON CONDITION CODE .NE.
BCR	141705	BR	-	-	SRV	BRANCH ON CBIT RESET.
BCS	141604	BR	-	-	SRV	BRANCH ON CBIT SET.
BDX	140734	BR	-	-	SRV	BRANCH ON DECREMENTED X.
BDY	140724	BR	-	-	SRV	BRANCH ON DECREMENTED Y.
BEQ	140612	BR	-	4	SRV	BRANCH ON A REGISTER .EQ. Ø.
BFEQ	50122	BR	-	4	I	BRANCH ON FLTG REG EQ.
BFEQ	141612	BR	-	4	SRV	BRANCH ON FAC .EQ. Ø.
BFGE	50125	BR	-	4	I	BRANCH ON FLTG REG NE.
BFGE	141615	BR	-	4	SRV	BRANCH ON FAC .GE. Ø.
BFGT	50121	BR	-	4	I	BRANCH ON FLTG REG LE.
BFGT	141611	BR	-	4	SRV	BRANCH ON FAC .GT. Ø.
BFLE	50120	BR	-	4	I	BRANCH ON FLTG REG LT.
BFLE	141610	BR	-	4	SRV	BRANCH ON FAC .LE. Ø.
BFLT	50124	BR	-	4	I	BRANCH on FLTG REG GT.
BFLT	141614	BR	-	4	SRV	BRANCH ON FAC .LT. Ø.
BFNE	50123	BR	-	4	I	BRANCH ON FLTG REG GE.
BFNE	141613	BR	-	4	SRV	BRANCH ON FAC .NE. Ø.
BGE	140615	BR	-	4	SRV	BRANCH ON A REGISTER .GE. Ø.
BGT	140611	BR	-	4	SRV	BRANCH ON A REGISTER .GT. Ø.
BHD1	50144	BR	-	-	I	BRANCH ON HALF REG DEC BY 1.
BHD2	50145	BR	-	-	I	BRANCH ON HALF REG DEC BY 2.
BHD4	50146	BR	-	-	I	BRANCH ON HALF REG DEC BY 4.
BHEQ	50112	BR	-	4	I	BRANCH ON HALF REG EQ.
BHGT	50111	BR	-	4	I	BRANCH ON HALF REG LE.
BHI1	50140	BR	-	-	I	BRANCH ON HALF REG INCR BY 1.
BHI2	50141	BR	-	-	I	BRANCH ON HALF REG INCR BY 2.
BHI4	50142	BR	-	-	I	BRANCH ON HALF REG INCR BY 4.
BHLE	50110	BR	-	4	I	BRANCH ON HALF REG LT.
BHLT	50104	BR	-	4	I	BRANCH ON HALF REG GT.
BHNE	50113	BR	-	4	I	BRANCH ON HALF REG GE.
BHNE	50105	BR	-	4	I	BRANCH ON HALF REG NE.
BIX	141334	BR	-	-	SRV	BRANCH ON INCREMENTED X.
BIY	141324	BR	-	-	SRV	BRANCH ON INCREMENTED Y.
BLE	140610	BR	-	4	SRV	BRANCH ON A REGISTER .LE. Ø.
BLEQ	140702	BR	-	4	SRV	BRANCH ON L REGISTER .EQ. Ø.
BLGE	140615	BR	-	4	SRV	BRANCH ON L REGISTER .GE. Ø.
BLGT	140701	BR	-	4	SRV	BRANCH ON L REGISTER .GT. Ø.
BLLE	140700	BR	-	4	SRV	BRANCH ON L REGISTER .LE. Ø.
BLLT	140614	BR	-	4	SRV	BRANCH ON L REGISTER .LT. Ø.
BLNE	140703	BR	-	4	SRV	BRANCH ON L REGISTER .NE. Ø.
BLR	141707	BR	-	-	SRV	BRANCH ON LINK RESET.
BLS	141706	BR	-	-	SRV	BRANCH ON LINK SET.
BLT	140614	BR	-	4	SRV	BRANCH ON A REGISTER .LT. Ø.
BMEQ	141602	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .EQ.
BMGE	141706	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .GE.
BMGT	141710	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .GT.
BMLE	141711	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .LE.
BMLT	141707	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .LT.
BMNE	141603	BR	-	-	SRV	BRANCH ON MAG.-COND. L,CC .NE.
BNE	140613	BR	-	4	SRV	BRANCH ON A REGISTER .NE. Ø.
BRBR	50040	BR	-	-	I	BRANCH ON REG BIT RESET.
BRBS	50000	BR	-	-	I	BRANCH ON REG BIT SET.
BGR1	50134	BR	-	-	I	BRANCH ON REG DEC BY 1.

BGR2	50135	BR	-	-	I	BRANCH ON REG DEC BY 2.
BGR4	50136	BR	-	-	I	BRANCH ON REG DEC BY 4.
BREQ	50102	BR	-	4	I	BRANCH ON REG EQ.
BRGE	50105	BR	-	4	I	BRANCH ON REG NE.
BRGT	50101	BR	-	4	I	BRANCH ON REG LE.
BRI1	50130	BR	-	-	I	BRANCH ON REG INCR BY 1.
BRI2	50131	BR	-	-	I	BRANCH ON REG INCR BY 2.
BRI4	50132	BR	-	-	I	BRANCH ON REG INCR BY 4.
BRLE	50100	BR	-	4	I	BRANCH ON REGISTER LT.
BRLT	50104	BR	-	4	I	BRANCH ON REGISTER GT.
BRNE	50103	BR	-	4	I	BRANCH ON REGISTER GE.

Control Operations

ARGT	000605	CON	-	-	SRV	ARG TRANSFER (USED WITH PCL).
HLT	000000	CON	-	-	SRVI	*HALT COMPUTER OPERATION.
IRTC	000603	CON	7	6	SRVI	*INTERRUPT RETURN, DO CAI.
IRTN	000601	CON	7	6	SRVI	*INTERRUPT RETURN, NO CAI.
ITLB	000615	CON	-	-	SRVI	*INVAL STLB ENTRY, L = VADDR.
LPID	000617	CON	-	-	SRVI	*LOAD PROCESS ID FROM A01-A12.
PRTN	000611	CON	7	6	SRVI	PROCEDURE RETURN.
SVC	000505	CON	-	-	SRVI	SUPERVISOR CALL.

Character String Operations

ZCM	001117	CS	X	X	VI	COMP TWO CHAR STR FIELDS.
ZFIL	011116	CS	X	X	VI	FILL CHAR STR FLD WITH CHAR.
ZMV	001114	CS	X	X	VI	CPY FRM SOURCE FLD TO DEST FLD.
ZTRN	001110	CS	X	X	VI	TRANS SRCE STR FLD TO DEST FLD.

Decimal Arithmetic

XAD	001100	DA	X	X	VI	ADD TWO DECIMAL FLDS.
XBTD	001145	DA	X	X	VI	CONV BI DEC VALUE TO DEC FLD.
XCM	001102	DA	X	X	VI	COMP TWO NUMERIC FLDS.
XDTB	001146	DA	X	X	VI	CONV DEC FLD TO BI REG VALUE.
XDZ	001107	DA	X	X	VI	DIV DEST FLD BY SOURCE FLD.
XMP	001104	DA	X	X	VI	MULTIPLY TWO DEC FLDS.
XMV	001101	DA	X	X	VI	MOVE NUM SOURCE FLD TO DST FLD.

Field and Edit Operations

XED	001112	FE	X	X	VI	EDIT NUMERIC FIELD.
ZED	001111	FE	X	X	VI	EDIT CHAR STR FIELD.

Field Operations

ALFA	001301	FLD	6	5	SRV	ADD L TO FIELD ADDR REG. ZERO.
ALFA	001311	FLD	6	5	SRV	ADD L TO FIELD ADDR REG. ONE.
ARFA	50161	FLD	-	7	I	ADD GR TO FIELD ADDR REG.
LDC	50162	FLD	-	7	I	LOAD CHAR TO GRH.
LDC	001312	FLD	-	7	SRV	LOAD CHAR TO A REG PER FAR 1.
LDC	001302	FLD	-	7	SRV	LOAD CHAR TO A REG VIA FAR Ø.
STC	50166	FLD	-	7	I	STORE CHARACTER FROM GRH.
STC	001332	FLD	-	7	SRV	STORE CHAR FROM A (SEE FAR1).
STC	001322	FLD	-	7	SRV	STORE CHAR FROM A (SEE FARØ).
TFLL	001333	FLD	-	-	SRV	XFER FLD LEN REG TO L REG 1.

```
TFLL ØØ1323 FLD - - SRV  XFER FLD LEN REG TO L REG Ø.
TFLR  5Ø163 FLD - 7 I    TRANSFER FLD LENGTH REG TO GR.
TLFL ØØ1321 FLD - - SRV  TRANS L TO FLD LEN REG Ø.
TLFL ØØ1331 FLD - - SRV  TRANSFER L TO FLD LEN REG 1.
TRFL  5Ø165 FLD - 7 I    TRANSFER GR TO FLD LENGTH REG.
```

Floating-point Operations

```
DBLE  5Ø1Ø6 FOP - - I    CONV SINGLE TO DOUBLE FLTG PT.
DFCM  5Ø144 FOP 3 1 I    DBL PRC FLTG COMP. -DFGR=>DFGR.
DFCM 14Ø574 FOP 3 5 SRV  -DFAC=>DFAC.
FCM   5Ø1ØØ FOP 3 1 I    FLTG COMP. -FGR=>FGR.
FCM  14Ø53Ø FOP 3 5 SRV  FLOATING COMP.  -FAC=>FAC.
FDBL 14ØØ16 FOP - - SRV  FAC=>DFAC.
FLTH  5Ø192 FOP - - I    // HALF WD INT TO FLTG PT.
FLOT 14Ø55Ø FOP 6 5 SRV  FLOT(A,B)=>FAC. (A,B) W/HOLE
FLT   5Ø1Ø5 FOP - - I    CONV INT TO FLTG PT FLOAT.
FLTA 14Ø532 FOP 3 5 SRV  FLOT(A)=>FAC.
FLTL 14Ø535 FOP 8 5 SRV  FLOT(L)=>FAC. (L W/NO HOLE)
FRN   5Ø1Ø7 FOP 3 1 I    FLOATING ROUND UP.
FRN  14Ø534 FOP 3 5 SRV  FLOATING ROUND UP.
INT   5Ø1Ø3 FOP - - I    INT(FRS)=>GR.
INT  14Ø554 FOP 3 5 SRV  INT(FAC)=>A,B W/HOLE.
INTA 14Ø531 FOP 3 5 SRV  INT(FAC)=>A.
INTH  5Ø1Ø1 FOP - - I    INT(FRS)=>GRH.
INTL 14Ø533 FOP 3 5 SRV  INT(FAC)=>L.
```

Floating-point Skip Operations

```
FSGT 14Ø515 FSK - 4 SRV  FLOATING SKIP IF .GT. Ø.
FSLE 14Ø514 FSK - 4 SRV  FLOATING SKIP IF .LE. Ø.
FSMI 14Ø512 FSK - 4 SRV  FLOATING SKIP IF .LT. Ø.
FSNZ 14Ø511 FSK - 4 SRV  FLOATING SKIP IF .NE. Ø.
FSPL 14Ø513 FSK - 4 SRV  FLOATING SKIP IF .GE. Ø.
FSZE 14Ø51Ø FSK - 4 SRV  FLOATING SKIP IF .EQ. Ø.
```

Integrity Operations

```
CXCS ØØ1714 IG  - - V    CONTROL EXTENDED CONTROL STORE.
EMCM ØØØ5Ø3 IG  - - SRVI*ENTER MACH CHK MODE.
LMCM ØØØ5Ø1 IG  - - SRVI*LEAVE MACHINE CHECK MODE.
LWCS ØØ171Ø IG  - - V    LOAD WRITABLE CONTROL STORE.
MDEI ØØ13Ø4 IG  - - SRVI*MEM DIAG ENABLE INTERLEAVE.
MDII ØØ13Ø5 IG  - - SRVI*MEM DIAGN INHIBIT INTERLEAVE.
MDIW OO13?? IG  - - SRVI*MEM DIAG WRT INTERLV.  L=>[E].
MDRS ØØ13Ø6 IG  - - SRVI*MEM DIAG READ SYNDROME BITS.
MDWC ØØ13Ø7 IG  - - SRV* MEM DIAG LOAD WRITE CTL REG.
RMC  ØØØØ21 IG  - - SRVI*RESET MACHINE CHECK FLAG.
VIRY ØØØ311 IG  5 6 SRV* EXECUTE VERIFICATON ROUTINE.
```

Input/Output Operations

```
CAI  ØØØ411 IO  - - SRVI*CLEAR ACTIVE INTERRUPT.
ENB  ØØØ4Ø1 IO  - - SRVI*ENABLE INTERRUPTS.
INH  ØØ1ØØ1 IO  - - SRVI*INHIBIT INTERRUPTS.
```

Logicize Operations

```
LCEQ  5Ø153 LOG - - I    IF .EQ., 1=>GRH, ELSE Ø=>GRH.
LCEQ 1415Ø3 LOG - - SRV  IF CC.EQ.,1=>A. ELSE Ø=>A.
LCGE  5Ø154 LOG - - I    IF .GE.,1=>GRH, ELSEØ=>GRH.
LCGE 1415Ø4 LOG - - SRV  IF CC.GE.,1=>A. ELSE Ø=>A.
LCGT  5Ø155 LOG - - I    IF .GT., 1=>GRH, ELSE Ø=>GRH.
LCGT 1415Ø5 LOG - - SRV  IF CC.GT.,1=>A. ELSE Ø=>A.
LCLE  5Ø151 LOG - - I    IF .LE., 1=>GRH, ELSE Ø=>GRH.
LCLE 1415Ø1 LOG - - SRV  IF CC.LE.,1=>A. ELSE Ø=>A.
LCLT  5Ø15Ø LOG - - I    IF .LT., 1=>GRH, ELSE Ø=>GRH.
LCLT 1415ØØ LOG - - SRV  IF CC.LT.,1=>A. ELSE Ø=>A.
LCNE  5Ø152 LOG - - I    IF .NE.,1=>GRH, ELSE Ø=>GRH.
LCNE 1415Ø2 LOG - - SRV  IF CC.NE.,1=>A. ELSE Ø=>A.
LEQ   5ØØØ3 LOG - 4 I    IF GR=Ø TN 1=>GRH,ELSE Ø=>GRH.
LEQ  14Ø413 LOG - 4 SRV  IF A.EQ.Ø, 1=>A. ELSE Ø=>A.
LF    5ØØ16 LOG - 4 I    LOGICIZE .F. Ø=>GRH.
LF   14Ø416 LOG - 5 SRV  LOGICIZE FALSE.  Ø=>A.
LFEQ  5ØØ23 LOG - 4 I    IF FRS=Ø TN1=>GRH,ELSE Ø=>GRH.
LFEQ 141113 LOG - 4 SRV  IF FAC.EQ.Ø, 1=>A. ELSE Ø=>A.
LFGE  5ØØ24 LOG - 4 I    IF FRS>=Ø T 1=>GRH,ELSE Ø=>GRH.
LFGE 141114 LOG - 4 SRV  IF FAC.GE.Ø, 1=>A. ELSE Ø=>A.
LFGT  5ØØ25 LOG - 4 I    IF FRS>Ø TN 1=>GRH,ELSE Ø=>GRH.
LFGT 141115 LOG - 4 SRV  IF FAC.GT.Ø, 1=>A. ELSE Ø=>A.
LFLE  5ØØ21 LOG - 4 I    IF FRS<=Ø T 1=>GRH,ELSE Ø=>GRH.
LFLE 141111 LOG - 4 SRV  IF FAC.LE.Ø, 1=>A. ELSE Ø=>A.
LFLT  5ØØ2Ø LOG - 4 I    IF FRS<Ø T 1=>GRH,ELSE Ø=>GRH.
LFLT 14111Ø LOG - 4 SRV  IF FAC.LT.Ø, 1=>A. ELSE Ø=>A.
LFNE  5ØØ22 LOG - 4 I    IF FRS<>Ø T 1=>GRH,ELSE Ø=>GRH.
LFNE 141112 LOG - 4 SRV  IF FAC.NE.Ø, 1=>A. ELSE Ø=>A.
LGE   5ØØØ4 LOG - 4 I    IF GR>=Ø TN 1=>GRH,ELSE Ø=>GRH.
LGE  14Ø414 LOG - 4 SRV  IF A.GE.Ø, 1=>A. ELSE Ø=>A.
LGT   5ØØØ5 LOG - 4 I    IF GR>Ø TN 1=>GRH, ELSE Ø=>GRH.
LGT  14Ø415 LOG - 4 SRV  IF A.GT.Ø, 1=>A. ELSE Ø=>A.
LHEQ  5ØØ13 LOG - 4 I    IF GRH=Ø TN 1=>GRH,ELSE Ø=>GRH.
LHGE  5ØØØ4 LOG - 4 I    IF GRH>=Ø T 1=>GRH,ELSE Ø=>GRH.
LHGT  5ØØ15 LOG - 4 I    IF GRH>Ø TN 1=>GRH,ELSE Ø=>GRH.
LHLE  5ØØ11 LOG - 4 I    IF GRH<=Ø T 1=>GRH,ELSE Ø=>GRH.
LHLT  5ØØØØ LOG - 4 I    IF GR<Ø TN 1=>GRH,ELSE Ø=>GRH.
LHNE  5ØØ12 LOG - 4 I    IF GRH<>Ø T 1=>GRH,ELSE Ø=>GRH.
LLE   5ØØØ1 LOG - 4 I    IF GR<=Ø TN 1=>GRH,ELSE Ø=>GRH.
LLE  14Ø411 LOG - 4 SRV  IF A.LE.Ø, 1=>A. ELSE Ø=>A.
LLEQ 141513 LOG - 4 SRV  IF L.EQ.Ø, 1=>A. ELSE Ø=>A.
LLGE 14Ø414 LOG - 4 SRV  IF L.GE.Ø, 1=>A. ELSE Ø=>A.
LLGT 141515 LOG - 4 SRV  IF L.GT.Ø, 1=>A. ELSE Ø=>A.
LLLE 141511 LOG - 4 SRV  IF L.LE.Ø, 1=>A. ELSE Ø=>A.
LLLT 14Ø41Ø LOG - 4 SRV  IF L.LT.Ø, 1=>A. ELSE Ø=>A.
LLNE 141512 LOG - 4 SRV  IF L.NE.Ø, 1=>A. ELSE Ø=>A.
LLT   5ØØØØ LOG - 4 I    IF GR<Ø TN 1=> GRH,ELSE Ø=>GRH.
LLT  14Ø41Ø LOG - 4 SRV  IF A.LT.Ø, 1=>A. ELSE Ø=>A.
LNE   5ØØØ2 LOG - 4 I    IF GR<>Ø TN 1=>GRH,ELSE Ø=>GRH.
LNE  14Ø412 LOG - 4 SRV  IF A.NE.Ø, 1=>A. ELSE Ø=>A.
LT   14Ø417 LOG - 5 SRV  LOGICIZE TRUE.  1=>A.
```

Mode Operations

```
DBL  ØØØØØ7 MOD - - SRV  ENTER DOUBLE-PREC MODE.
E16S ØØØØ11 MOD - - SRV  ENTER P3ØØ 16K SECTORED MODE.
```

E32I	ØØ1Ø1Ø	MOD	-	-	SRV	ENTER P5ØØ 321 MODE.
E32R	ØØ1Ø13	MOD	-	-	SRVI	ENTER P3ØØ 32K RELATIVE MODE.
E32S	ØØØØ13	MOD	-	-	SRVI	ENTER P3ØØ 32K SECTORED MODE.
E64R	ØØ1Ø11	MOD	-	-	SRVI	ENTER P3ØØ 64K RELATIVE MODE.
E64V	ØØØØ1Ø	MOD	-	-	SRVI	ENTER P4ØØ MODE.
ESIM	ØØØ415	MOD	-	-	SRVI*	ENTER STANDAGR INTERRUPT MODE.
EVIM	ØØØ417	MOD	-	-	SRVI*	ENTER VECTORED INTERUPT MODE.
SGL	ØØØØØ5	MOD	-	-	SRV	ENTER SINGLE-PRECISION MODE.

Memory-reference Operations

ADD	Ø6	MR	2	1	SRV	ADD. (A) + [EA]16 => A.
ADL	Ø6 Ø3	MR	2	1	V	(A,B)+[EA]32=>A,B. (NO HOLE).
AH	12	MR	2	1	I	ADD HALFWD. RH+[EA16]=>RH.
ANA	Ø3	MR	-	-	SRV	AND. (A) .AND. [EA]16 => A.
ANL	Ø3 Ø3	MR	-	-	V	(A,B) .AND. [EA]32 => A,B.
C	61	MR	1	1	I	COMPARE R WITH [EA32].
CAS	11	MR	1	1	SRV	SKIP Ø,1,2 IF (A) >,=.< [EA]16.
CH	71	MR	1	1	I	COMPARE RH WITH [EA16].
CLS	11 Ø3	MR	1	1	V	SKIP Ø,1,2 IF (A,B)>,=.<[EA]32.
CREP	1Ø Ø2	MR	-	-	R	(P) => [(S)+1]16, EA=>P.
D	62	MR	3	1	I	DIV. (R,R+1)/[EA32=>R RMR=>R+1.
DAD	Ø6(DP)	MR	2	1	SR	(A,B)+[EA]32 => A,B.(W/HOLE).
DFA	15 17	MR	3	1	I	DBLE FLTG ADD. DFR+[EA64]=>DFR.
DFAD	Ø6 Ø2	MR	3	5	RV	(DFAC) + [EA]64 => DFAC.
DFC	Ø5 Ø7	MR	-	1	I	DBLE FLTG COMP DRF TO [EA64].
DFCS	11 Ø2	MR	6	5	RV	SKP Ø,1,2 IF (DFAC)>,=.<[EA]64.
DFD	31,33	MR	3	1	I	DOUBLE FLTG DIVIDE.
DFDV	17 Ø2	MR	3	5	RV	(DFAC) / [EA]64 => DFAC.
DFL	Ø1 Ø3	MR	-	-	I	DBLE FLTG LOAD. [EA643]=>DFR.
DFLD	Ø2 Ø2	MR	-	-	RV	[EA]64 => DFAC.
DFLX	15 Ø2	MR	-	-	V	LD DFLT INDEX. 4*[EA]16 => X.
DFM	25 27	MR	3	1	I	DBL FLTG MULT. DFR/[EA64]=>DFR.
DFMP	16 Ø2	MR	3	5	RV	(DFAC) * [EA]64 => DFAC.
DFS	21 23	MR	3	1	I	DBLE FLTG SUB. DFR-[EA64]=>DFR.
DFSB	Ø7 Ø2	MR	3	5	RV	(DFAC) - [EA]64 => DFAC.
DFST	11,13	MR	1	1	I	DBLE FLTG STORE. DFR=>[EA64].
DFST	Ø4 Ø2	MR	-	-	RV	(DFAC) => [EA]64.
DH	72	MR	3	1	I	DIV HW. R/[EA16]=>RH RM=>RL(2).
DIV	17	MR	3	5	V	(A,B)/[EA]16=>A,REM=>B.(NOHOLE)
DIV	17	MR	3	5	SR	(A,B)/[EA]16=>A;REM=>B.(W/HOLE)
DLD	Ø2(DP)	MR	-	-	SR	DOUBLE LOAD. [EA]32 => A,B.
DM	6Ø	MR	-	1	I	DECR. [EA32]-1=>[EA32].
DMH	7Ø	MR	-	1	I	DECR HALDWD. [EA16]-1=>[EA16].
DSB	Ø7(DP)	MR	2	1	SR	(A,B)-[EA]32 => A,B (W/HOLE).
DST	Ø4(DP)	MR	-	-	SR	DOUBLE STORE. (A,B) => [EA]32.
DVL	17 Ø3	MR	3	5	V	(A,B,E)/[EA]32=>A,B REM=>EH,EL.
EAA	Ø1 Ø1	MR	-	-	R	EFF. ADDR TO A-REG. EA => A.
EAL	Ø1 Ø1	MR	-	-	V	LOAD EFFECTIVE ADDR. EA => L.
EALB	42	MR	-	-	I	EFF ADDR to LB. EA=>LB.
EALB	13 Ø2	MR	-	-	V	EFF. ADDR TO LB. EA => LB.
EAR	63	MR	-	-	I	EFF ADDR (EA32)TO R.
EAXB	52	MR	-	-	I	EFF ADDR TO XB. EA=>XB.
EAXB	12 Ø2	MR	-	-	V	EFF. ADDR TO XB. EA => XB.
EIO	34	MR	-	2	I	EXECUTE ADDR TO BASE REG.
EIO	14 Ø1	MR	-	7	V*	EXEC EA AS I/O INSTR.CCEQ=>SKP.
ENTR	Ø1 Ø3	MR	-	-	R	(S)=>[(S)-EA]16, (S)-EA=>S.

ERA	Ø5	MR	-	-	SRV	(A) .XOR. [EA]16 => A.
ERL	Ø5 Ø3	MR	-	-	V	(A,B) .XOR. [EA]32 => A,B.
FA	14,16	MR	3	1	I	FLTG ADD. FR+[EA32]=>FR.
FAD	Ø6 Ø1	MR	3	5	RV	(FAC) + [EA]32 => FAC.
FC	Ø4,Ø6	MR	-	1	I	FLTG COMPARE FR TO [EA32]
FCS	11 Ø1	MR	6	5	RV	SKIP Ø,1,2 IF (FAC)>,=.<[EA]32.
FD	3Ø,32	MR	3	1	I	FLTG DIV. -FR/[EA32]=>FR.
FDV	17 Ø1	MR	3	5	RV	(FAC) / [EA]32 => FAC.
FLD	Ø2 Ø1	MR	-	-	RV	FLOATING LOAD. [EA]32 => FAC.
FLX	15 Ø1	MR	-	-	RV	2*[EA]16 => X.
FM	24,26	MR	3	1	I	FLTG MULT. FR*[EA32]=>FR.
FMP	16 Ø1	MR	3	5	RV	(FAC) * [EA]32 => FAC.
FS	2Ø,22	MR	3	1	I	FLTG SUB. FR-[EA32]=>FR.
FSB	Ø7 Ø1	MR	3	5	RV	(FAC) - [EA]32 => FAC.
FST	1Ø,12	MR	-	-	I	FLTG STORE. FR=>[EA32].
FST	Ø4 Ø1	MR	6	6	RV	(FAC) => [EA]32.
I	41	MR	-	-	I	INTERCHANGE R WITH [EA32].
IH	51	MR	-	-	I	INTERCHANGE RH WITH [EA16].
IM	4Ø	MR	-	1	I	INCR. [EA32]+1=>[EA32].
IMA	13	MR	-	-	SRV	EXCHANGE MEMORY AND A-REGISTER.
IMH	Ø	MR	-	1	I	INCR HALD WD> [EA16]+1=>[EA16].
IRS	12	MR	-	-	SRV	INC, REPLACE, AND SKIP IF ZERO.
JDX	15 Ø2	MR	-	-	R	JUMP ON DECREMENTED X-R ZERO.
JEQ	Ø2 Ø3	MR	-	-	R	IF (A) .EQ. Ø, EA => P.
JGE	Ø7 Ø3	MR	-	-	R	IF (A) .GE. Ø, EA => P.
JGT	Ø5 Ø3	MR	-	-	R	IF (A) .GT. Ø, EA => P.
JIX	15 Ø3	MR	-	-	R	JUMP ON INCREMENTED X-REG ZERO.
JLE	Ø4 Ø3	MR	-	-	R	IF (A) .LE. Ø, EA => P.
JLT	Ø6 Ø3	MR	-	-	R	IF (A) .LT. Ø, EA => P.
JMP	51	MR	-	-	I	JUMP. EA=>RP.
JMP	Ø1	MR	-	-	SRV	UNCOND JUMP. EA => PB,P.
JNE	Ø3 Ø3	MR	-	-	R	IF (A) .NE. Ø, EA => P.
JSR	73	MR	-	-	I	RPL=>RH, EA32=>RP
JST	1Ø	MR	-	-	SRV	(P) => [EA]16, EA+1 => P.
JSX	35 Ø3	MR	-	-	RV	(P) => X, EA => P.
JSXB	61	MR	-	-	I	RP=>XB, EA=>RP.
JSXB	14 Ø2	MR	-	-	V	(PB,P) => XB, EA => PB,P.
JSY	14	MR	-	-	V	(P) => Y, EA => P.
L	Ø1	MR	-	-	I	LOAD. [EA32]=>R.
LDA	Ø2	MR	-	-	SRV	LOAD A-REGISTER. [EA]16 => A.
LDAR	44	MR	-	-	I	LOAD ADDR REGISTER.
LDL	Ø2 Ø3	MR	-	-	V	LOAD LONG. [EA]32 => A,B.
LDLR	Ø5 Ø1	MR	-	-	V	LOAD LONG FROM RFILE LOC EA.
LDX	35	MR	-	-	SRV	LOAD X-REGISTER. [EA]16 => X.
LDY	35 Ø1	MR	-	-	V	LOAD Y-REGISTER. [EA]16 => Y.
LH	11	MR	-	-	I	LOAD HALF WD. [EA16]=>RH.
LHL1	Ø4	MR	-	-	I	SHIFTED 1 [EA16] .LS. 1=>GRH.
LHL2	14	MR	-	-	I	SHIFTED 2 [EA16] .LS. 2=>GRH.
M	42	MR	3	1	I	MULT. R*[EA32]=>(R,R+1).
MH	52	MR	3	1	I	RH*[EA16}=>R.
MIA	64	MR	-	-	I	MICROCODE EXECUTE A.
MIA	12 Ø1	MR	-	-	V	MICROCODE ENTRANCE.
MIB	74	MR	-	-	I	MICROCODE EXECUTE B.
MIB	13 Ø1	MR	-	-	V	MICROCODE ENTRANCE.
MPL	16 Ø3	MR	-	1	V	(A,B) * [EA]16 => A,B.(NOHOLE)
MPY	16	MR	-	1	V	(A) * [EA]16 => A,B. (NOHOLE).
MPY	16	MR	3	1	SR	(A) * [EA]16 => A,B. (HOLE).

N	Ø3	MR	-	-	I	R .AND. [EA32]=>R.
NH	13	MR	-	-	I	RH .AND. [EA16]=>RH.
O	23	MR	-	-	I	R .OR. [EA32]=>R.
OH	33	MR	-	-	I	RH .OR. [EA16]=>RH.
ORA	Ø3 Ø2	MR	-	-	V	(A) .OR. [EA]16 => A.
PCL	41	MR	-	-	I	PROCEDURE CALL.
PCL	1Ø Ø2	MR	7	6	V	P-4ØØ PROCEDURE CALL.
S	22	MR	2	1	I	SUB. R-[EA32]=>R.
SBL	Ø7 Ø3	MR	2	1	V	(A,B) - [EA]32 => A,B.(NOHOLE)
SH	32	MR	2	1	I	SUB HALF WD. RH-[EA16]=>RH.
SHA	15	MR	4	-	I	ARITHMETIC SHIFT.
SHL	Ø5	MR	4	-	I	LOGICAL SHIFT.
ST	21	MR	-	-	I	STORE. R=>[EA32].
STA	Ø4	MR	-	-	SRV	STORE A-REG. (A) => [EA]16.
STAR	54	MR	-	-	I	STORE ADDRESSED REGISTER.
STH	31	MR	-	-	I	STORE HALF WD. RH=>[EA16].
STL	Ø4 Ø3	MR	-	-	V	STORE LONG. (A,B) => [EA]32.
STLR	Ø3 Ø1	MR	-	-	V	STORE LONG INTO RFILE LOC EA.
STX	15	MR	-	-	SRV	STORE X-REG. (X) => [EA]16.
STY	35 Ø2	MR	-	-	V	STORE Y-REG. (Y) => [EA]16.
SUB	Ø7	MR	2	1	SRV	SUBTRACT. (A) - [EA]16 => A.
TM	44	MR	-	1	I	REST MEM. [EA32]::Ø=>CC.
X	43	MR	-	-	I	EXC OR. R .XOR. [EA32]=>R.
XEC	Ø1 Ø2	MR	-	-	RV	EXECUTE INSTRUCTION AT EA.
XEC	Ø1 Ø2	MR	-	-	RV	EX EFF ADDR CONT AS THIS INST.
XH	52	MR	-	-	I	RH .XOR. [EA16]=>RH.
ZM	43	MR	-	-	I	ZERO MEM. Ø=>[EA32].
ZMH	53	MR	-	-	I	ZERO MEM. HALF WD. Ø=>[EA16].

Miscellaneous Operations

A1A	1412Ø6	OPR	2	1	SRV	ADD 1 TO A REG. A+1=>A.
A2A	14Ø3Ø4	OPR	2	1	SRV	ADD 2 TO A REGISTER. A+2=>A.
ABQ	5Ø134	OPR	-	7	I	ADD TO BOTTOM OF QUEUE.
ACA	141216	OPR	2	1	SRV	ADD CBIT TO A REG. CBIT+A=>A.
ADLL	141ØØØ	OPR	2	1	SRV	ADD LINK TO L REGISTER.
ADLR	5ØØ14	OPR	-	7	I	ADD LINK TO GR.
ATQ	5Ø135	OPR	-	7	I	ADD TO TOP OF QUEUE.
CAL	141Ø5Ø	OPR	-	-	SRV	CLEAR A REG. LEFT BYTE.
CAR	141Ø44	OPR	-	-	SRV	CLEAR A REG. RIGHT BYTE.
CAZ	14Ø214	OPR	1	1	SRV	SKIP Ø,1,2 INST. IF A >,=,< Ø.
CEA	ØØØ111	OPR	-	-	SRV	A AS EA=>A. (USELESS IN 64V).
CGT	5ØØ26	OPR	-	7	I	COMPUTED GOTO.
CHS	5ØØ4Ø	OPR	-	-	I	CHANGE SIGN -(1)=>GR(1).
CHS	14ØØ24	OPR	-	-	SRV	CHANGE SIGN OF A REGISTER.
CMA	14Ø4Ø1	OPR	-	-	SRV	ONE'S COMPLEMENT A REGISTER.
CMH	5ØØ45	OPR	-	-	I	COMP HALF REG. GRH=>GRH.
CMR	5ØØ44	OPR	-	-	I	COMP REG. GR=>GR.
CR	5ØØ56	OPR	-	-	I	CLEAR REG. Ø => GR.
CRA	14ØØ4Ø	OPR	-	-	SRV	CLEAR A REGISTER. Ø=>A.
CRB	14ØØ15	OPR	-	-	SRV	CLEAR B REGISTER. Ø=>B.
CRBL	5ØØ62	OPR	-	-	I	LEFT BYTE Ø=>GRH(1-8).
CRBR	5ØØ63	OPR	-	-	I	RIGHT BYTE Ø=>GRH(9-16).
CRE	1414Ø4	OPR	-	-	SRV	CLEAR E. Ø=>E.
CRL	14ØØ1Ø	OPR	-	-	SRV	CLEAR L REGISTER. Ø=>L.
CRLE	14141Ø	OPR	-	-	SRV	CLEAR L AND E. Ø=>L, Ø=>E.
CSA	14Ø32Ø	OPR	5	-	SRV	CPY SIGN OF A. A1=>CBIT,Ø=>A1.

CSR	5ØØ41	OPR	-	-	I	COPY & SAVE SIGN. 1=>C,Ø=>GR1.
DH1	5Ø13Ø	OPR	2	1	I	DECR HALF REG BY 1. GRH-1=>GRH.
DH2	5Ø131	OPR	2	1	I	DECR HALF REG BY 2. GRH-2=>GRH.
DR1	5Ø124	OPR	2	1	I	DECR REG BY 1. GR-1=>GR.
DR2	5Ø125	OPR	2	1	I	DECR REG BY 2. GR-2=>GR.
DRX	14Ø21Ø	OPR	-	-	SRV	DECREMENT X AND SKIP IF Ø.
IAB	ØØØ2Ø1	OPR	-	-	SRV	EXCHANGE A AND B A=>B & B=>A.
ICA	14134Ø	OPR	-	-	SRV	INTERCHANGE BYTES OF A REG.
ICBL	5ØØ65	OPR	-	-	I	GRH(1-8)=>GRH(9-16), Ø=>R.
ICBR	5ØØ66	OPR	-	-	I	GRH(9-16)=>GRH(1-8), Ø=>
ICHL	5ØØ6Ø	OPR	-	-	I	GRH=>GRL, Ø=>GRH.
ICHR	5ØØ61	OPR	-	-	I	GRL=>GRH, Ø=>GRL.
ICL	14114Ø	OPR	-	-	SRV	EXCHANGE BYTES OF A, CLR LEFT.
ICR	14124Ø	OPR	-	-	SRV	EXCHANGE BYTES OF A, CLR RIGHT.
IH1	5Ø126	OPR	2	1	I	GRH+1=>GRH.
IH2	5Ø127	OPR	2	1	I	GRH+2=>GRH.
ILE	141414	OPR	-	-	SRV	EXCHANGE L AND E. L=>E & E=>L.
INK	5ØØ7Ø	OPR	-	-	I	MOVE KEYS TO REG KEYS.
INK	ØØØØ43	OPR	-	-	SRV	INPUT P-3ØØ KEYS INTO A REG.
IR1	5Ø122	OPR	2	1	I	INCR REG BY 1. GR+1=>GR.
IR2	5Ø123	OPR	2	1	I	INCR REG BY 2. GR+2=>GR.
IRB	5ØØ62	OPR	-	-	I	BYTES GRH(1-8)=>GRH(9-16)
IRH	5ØØ57	OPR	-	-	I	GRH=>GRL, GRL=>GRH.
IRX	14Ø114	OPR	-	-	SRV	INCREMENT X AND SKIP IF Ø.
NOP	ØØØØØ1	OPR	-	-	SRV	NO OPERATION.
NRM	ØØØ1Ø1	OPR	-	-	SRV	NORMALIZE A,B AS ON P-3ØØ.
OTK	5ØØ71	OPR	7	S	I	SET KEYS FROM REGISTER
OTK	ØØØ4Ø5	OPR	7	6	SRV	OUTPUT A TO 3ØØ KEYS, SHFT CTR.
PID	5ØØ52	OPR	-	-	I	GR=>GRN, (1)=>GR(2-32).
PID	ØØØ211	OPR	-	-	SRV	SHORT INT TO DP INT W/HOLE.
PIDA	ØØØ115	OPR	-	-	SRV	SHORT TO LONG INT CONV. A=>L.
PIDH	5ØØ53	OPR	-	-	I	GRH=>GRL,GRH(1)=>GRH(2-16).
PIDL	ØØØ3Ø5	OPR	-	-	SRV	CONVERT LONG INT TO 64 BIT INT.
PIM	5ØØ5Ø	OPR	2	1	I	POS REG AFTER INT MULTIPLY.
PIM	ØØØ2Ø5	OPR	-	-	SRV	DP INT WITH HOLE TO SHORT INT.
PIMA	ØØØØ15	OPR	3	5	SRV	L=>A. IEX ON PREC LOSS.
PIMH	5ØØ51	OPR	2	1	I	POS HALF REG AFTER INT MULT.
PIML	ØØØ3Ø1	OPR	3	5	SRV	64BIT INT TO LONG INT.(L,E)=>L.
RBQ	5Ø133	OPR	-	7	I	REMOVE FROM BOTTOM OF QUEUE.
RCB	14Ø2ØØ	OPR	5	-	SRV	RESET CBIT. Ø=>CBIT.
RTN	ØØØ1Ø5	OPR	-	-	SRV	RETURN FROM P-3ØØ RECUR PROC.
RTQ	5Ø132	OPR	-	7	I	REMOVE FROM TOP OF QUEUE.
S1A	14Ø11Ø	OPR	2	1	SRV	SUB 1 FROM A REGISTER. A-1=>A.
S2A	14Ø31Ø	OPR	2	1	SRV	SUB 2 FROM A REGISTER. A-2=>A.
SCA	ØØØØ41	OPR	-	-	SRV	LOAD P-3ØØ SHFT CTR INTO A REG.
SCB	14Ø6ØØ	OPR	5	-	SRV	SET CBIT. 1=>CBIT.
SHL1	5ØØ76	OPR	4	1	I	GRH .LS. 1 => GRH.
SHL2	5ØØ77	OPR	4	1	I	GRH .LS. 2 => GRH.
SHR1	5Ø12Ø	OPR	4	1	I	GRH .RS. 1 => GRH.
SHR2	5Ø121	OPR	4	1	I	GRH .RS. 2 => GRH.
SL1	5ØØ72	OPR	4	1	I	GR .LS. 1 => GR.
SL2	5ØØ73	OPR	4	1	I	GR .LS. 2 => GR.
SR1	5ØØ74	OPR	4	1	I	GR .RS. 1 => GR.
SR2	5ØØ75	OPR	4	1	I	GR .RS. 2 => GR.
SSM	5ØØ42	OPR	-	-	I	SET SIGN. MINUS 1=>(1).
SSM	14Ø5ØØ	OPR	-	-	SRV	SET SIGN OF A MINUS. 1=>A1.
SSP	5ØØ43	OPR	-	-	I	SET SIGN. PLUS Ø=>(1).

SSP	14Ø1ØØ	OPR	-	-	SRV	SET SIGN OF A PLUS. Ø=>ABIT1.
STCD	5Ø137	OPR	-	7	I	GR+1=[EA16].
STCH	5Ø136	OPR	-	7	I	GRL=[EA16].
STEX	5ØØ27	OPR	-	7	I	STACK EXTEND.
STEX	ØØ1315	OPR	6	5	SRV	STK EXTEND. L REG HAS EXTENT.
TAB	14Ø314	OPR	-	-	SRV	TRANSFER A TO B REG. A=>B.
TAK	ØØ1Ø15	OPR	7	6	SRV	TRANSFER A TO KEYS.
TAX	14Ø5Ø4	OPR	-	-	SRV	TRANS A REG TO X REG. A=>X.
TAY	14Ø5Ø5	OPR	-	-	SRV	TRANS A REG TO Y REG. A=>Y.
TBA	14Ø6Ø4	OPR	-	-	SRV	TRANS B REG TO A REG. B=>A.
TC	5ØØ47	OPR	3	1	I	-GR+1=>GR.
TCA	14Ø4Ø7	OPR	2	1	SRV	TWO'S COMPLEMENT A. -A=>A.
TCH	5ØØ47	OPR	3	1	I	-GRH+1=>GRH.
TCL	14121Ø	OPR	2	1	SRV	TWO'S COMPLEMENT L. -L=>L.
TKA	ØØ1ØØ5	OPR	-	-	SRV	TRANSFER KEYS TO A.
TSTQ	5Ø1Ø4	OPR	-	7	I	TEST QUEUE.
TXA	141Ø34	OPR	-	-	SRV	TRANS X REG TO A REG. X=>A.
TYA	141124	OPR	-	-	SRV	TRANS Y REG TO A REG. Y=>A.
XCA	14Ø1Ø4	OPR	-	-	SRV	EXCHG AND CLR A. A=>B, Ø=>A.
XCB	14Ø2Ø4	OPR	-	-	SRV	EXCHG AND CLR B. B=>A, Ø=>B.
xxx	14ØØ14	OPR	-	-	SRV	OBSOLETE. CLRS B, LSW OF DFAC.

Programmed I/O Operations

INA	54	PIO	-	-	SR*	INPUT TO A-REGISTER.
OCP	14	PIO	-	-	SR*	OUTPUT CONTROL PULSE.
OTA	74	PIO	-	-	SR*	OUTPUT FROM A-REGISTER.
SKS	34	PIO	-	-	SR	SKIP IF CONDITION SET.
SMK	17ØØ2Ø	PIO	-	-	SR*	SET INTERRUPT MASKS.

Shift Operations

ALL	Ø414XX	SH	4	5	SRV	A LEFT LOGICAL.
ALR	Ø416XX	SH	4	5	SRV	A LEFT ROTATE.
ALS	Ø415XX	SH	4	5	SRV	A LEFT SHIFT (SHORT INT ARITH).
ARL	Ø4Ø4XX	SH	4	5	SRV	A RIGHT LOGICAL.
ARR	Ø4Ø6XX	SH	4	5	SRV	A RIGHT ROTATE.
ARS	Ø4Ø5XX	SH	4	5	SRV	A RIGHT SHIFT (SHORT ARITH).
LLL	Ø41ØXX	SH	4	5	SRV	LONG LEFT LOGICAL.
LLR	Ø412XX	SH	4	5	SRV	LONG LEFT ROTATE.
LLS	Ø411XX	SH	4	5	SRV	LNG LSFT 64V=>LNG INT,ELSE HOLE
LRL	Ø4ØØXX	SH	4	5	SRV	LONG RIGHT LOGICAL.
LRR	Ø4Ø2XX	SH	4	5	SRV	LONG RIGHT ROTATE.
LRS	Ø4Ø1XX	SH	4	5	SRV	LNG RSFT.64V=LNG INT,ELSE HOLE.

Skip Operations

SAR	1ØØ26X	SKP	-	-	SRV	SKIP IF A REG. BIT N RESET.
SAS	10126Ø	SKP	-	-	SRV	SKIP IF A REG. BIT N SET.
SGT	1ØØ22Ø	SKP	-	-	SRV	SKIP IF A REG. .GT. Ø.
SKP	1ØØØØØ	SKP	-	-	SRV	SKIP ONE WORD.
SLE	1Ø122Ø	SKP	-	-	SRV	SKIP IF A REG. .LE. Ø.
SLN	1Ø11ØØ	SKP	-	-	SRV	SKIP IF A REG. BIT 16 SET.
SLZ	1ØØ1ØØ	SKP	-	-	SRV	SKIP IF A REG. BIT 16 .EQ. Ø.
SMCR	1ØØ2ØØ	SKP	-	-	SRV	SKIP IF MACHINE CHECK RESET.
SMCS	1Ø12ØØ	SKP	-	-	SRV	SKIP IF MACHINE CHECK SET.
SMI	1Ø14ØØ	SKP	-	-	SRV	SKIP IF A REG. .LT. Ø.

SNR	1ØØ24X	SKP	-	-	SRV*	SKIP IF SENSE SWITCH N RESET.
SNS	1Ø124Ø	SKP	-	-	SRV*	SKIP IF SENSE SWITCH N SET.
SNZ	1Ø1Ø4Ø	SKP	-	-	SRV	SKIP IF A REG. NE. Ø.
SPL	1ØØ4ØØ	SKP	-	-	SRV	SKIP IF A REG. .GE. Ø.
SR1	1ØØØ2Ø	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 1 RESET.
SR2	1ØØØ1Ø	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 2 RESET.
SR3	1ØØØØ4	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 3 RESET.
SR4	1ØØØØ2	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 4 RESET.
SRC	1ØØØØ1	SKP	-	-	SRV	SKIP IF CBIT RESET.
SS1	1Ø1Ø2Ø	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 1 SET.
SS2	1Ø1Ø1Ø	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 2 SET.
SS3	1Ø1ØØ4	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 3 SET.
SS4	1Ø1ØØ2	SKP	-	-	SRV*	SKIP IF SENSE SWITCH 4 SET.
SSC	1Ø1ØØ1	SKP	-	-	SRV	SKIP IF CBIT SET.
SSR	1ØØØ36	SKP	-	-	SRV*	SKIP IF SSWI 1,2,3 AND 4 RESET.
SSS	1Ø1Ø36	SKP	-	-	SRV*	SKIP IF SSWI 1,2,3 AND 4 SET.
SZE	1ØØØ4Ø	SKP	-	-	SRV	SKIP IF A REG .EQ. Ø.

Virtual-memory Operations

EPMJ	ØØØ217	VM	-	-	SR	ENT PAGE MODE AND JUMP (P3ØØ).
EPMX	ØØØ237	VM	-	-	SR	ENT PAG MOD & JMP TO MICROCODE.
ERMJ	ØØØ7Ø1	VM	-	-	SR	ENTER RESTR MODE & JUMP (P3ØØ).
ERMX	ØØØ721	VM	-	-	SR	RESTR MOD & JUMP TO MICROCODE.
EVMJ	ØØØ7Ø3	VM	-	-	SR	ENT VIRT MODE AND JUMP (P3ØØ).
EVMX	ØØØ723	VM	-	-	SR	VIRT MOD & JUMP TO MICROCODE.
LPMJ	ØØØ215	VM	-	-	SR	LEAVE PAGE MODE & JUMP (P3ØØ).
LPMX	ØØØ235	VM	-	-	SR	LVE PAG MOD & JMP TO MICROCODE.

6 OPERATIONAL PROCEDURES

BOOT PROCEDURES

Master clear, select LOAD, raise data switches 1-16 as follows ('-' => don't care):

```
 1                 16
|A|AAA|AAA|AAA|---|000| Start at 'AAAAAAAAAA000000
|S|SSS|SSS|DDP|---|001| ASR Paper Tape
|S|SSS|SSS|DDP|---|010| High Speed Paper Tape
|H|HHH|HHH|H--|--0|011| Option B  FHD
|H|HHH|HHH|H--|-01|011| Option B' FHD, DA='21
|H|HHH|HHH|H--|-11|011| Option B' FHD, DA='23
|H|HHH|HHH|H--|0-0|100| Option B  Upper MHD
|H|HHH|HHH|H--|1-0|100| Option B  Lower MHD
|H|HHH|HHH|H-0|001|100| Option B' Upper MHD, DA='21
|H|HHH|HHH|H-0|101|100| Option B' Lower MHD, DA='21
|H|HHH|HHH|H-0|011|100| Option B' Upper MHD, DA='23
|H|HHH|HHH|H-0|111|100| Option B' Lower MHD, DA='23
|H|HHH|HHH|H-1|-01|100| Storage Module, DA='26
|H|HHH|HHH|H-1|-11|100| Storage Module, DA='27
|N|NNN|NNN|RRS|CT-|101| Magtape
|-|---|---|---|---|110| Diskette (Floppy)
|-|---|---|---|---|111| Unused
```

A..A Addr/'100 to start at
S..S Sector for boot loader relocation
DD Displacement for boot loader relocation
P 1 => suppress auto-start
C 1 => halt to allow baud rate change
H..H DOS select:
00000000 Highest that will fit
010----- *DOS16
011----- *DOS64
100----- *DOS32
N..N File number to load (0=>prompts)
RR Relocation of boot program to ending address of:
00 - end of physical memory
01 - 16K
10 - 32K
11 - 48K
T 0 - 9-track
1 - 7-track

BOOT TERMINAL SPEED SELECTION

Parameter 4 (B Register)
'110 for 110 BAUD
'1010 for 300 BAUD
'2010 for 1200 BAUD
'3410 for 9600 BAUD

Parameter 5 (X Register)
'27 for 110 BAUD
'76 for 300 BAUD
'373 for 1200 BAUD
'3735 for 9600 BAUD

Parameter 6 (Keys)
'740** for 110 BAUD
'340** for 1200 - 9600 BAUD

** = number of delays after .CR.

TYPICAL SWITCH SETTINGS FOR DISK BOOTS

DVNO	Switches
0	--0004
1	--0044
10	--0003
20	--0006
30	--0014
31	--0054
40	--0014
---050	--0014
---051	--0054
---250	--0034
---251	--0074
---460	--0114

Hit START to initiate load sequence. Select RUN after load has started.

COLD START (PRIMOS IV,V)

1) Boot in PRIMOS II (see 'BOOT' above), startup disk containing UFD with PRIMOS IV (typically PR4.64, PR4.16, PR4L16). Attach to UFD containing PRIMOS IV.

2) Type 'R PRIMOS'. If there is a C_PRMO file in the current CMDNC0 the configuration will be taken from that file. Else enter CONFIG command:

CONFIG -- SET SYSTEM CONFIGURATION PARAMETERS

```
CONFIG [<node>] <ntusr> <pagdev> <comdev>
 [<memsiz>] [<altdev>] [<namlc>] [<npusr>]
 [<nrusr>] [<slmcon>]

                    -or-

CONFIG -DATA <config-filename>
```

Commands for latter form documented in PE-T-412.

Cold start command only.

3) If PRIMOS IV halts at 1507 (1510 in address lights), it has encountered bad memory during initial memory scan (see HALTS). Hit start to map out the bad page and continue.

4) After the introductory messages, enter the date and time:

```
SEtime -<mmddyy> -<hhmm>
```

Users may now log into the system.

HALTS

ON MACHINE HALT:

1) Select STOP/STEP mode (rotary switch).
2) Place ADDRESS/DATA switch on ADDRESS and note the address displayed in lights. To determine the segment number of the halt, select FETCH, depress DATA CLEAR, set data switches to '14, raise switches 1 and 4, depress START, note displayed segment number.
3) Refer to latest load map of PRIMOS IV and/or the following list of load map entries to determine subsequent actions. (NOTE: the address displayed at the halt will always be one location higher than the corresponding halt location address.)

PRIMOS IV HALT LOCATIONS

(Addresses marked with a '+' are those most likely to change when PRIMOS is reloaded. The letters in parentheses refer to subsequent actions that should be taken and are described following the list.)

AMLCI_	+6/12714	Bad AMLC Interrupt.	(D,W)
BDMEM_	4/1507	Parity error during cold start.	(M,C)
BOOT0_	+6/10634	Halt after SHUTDN ALL command.	(C)
IFLTB_	+4/115204	Fault in interrupt handler.	(D,C)
INTRT_	+4/115357	Too many PRTNs.	(D,C)
IPAGE_	+4/115321	Page flt in interrupt process.	(D,C)
MCHK_	4/305	Machine check.	(D,W)
MEMH2_	4/317	Halt after mapout of bad page.	(W*)
MEMPA_	4/276	Uncorrected mem parity error.	(M,W*)
MMOD_	4/315	Missing memory module.	(D,C)
PAGFB_	6/3577	Illegal page fault.	(D,C)
REFL0_	6/3663	Illegal FLEX, UII, PSU.	(D,C)
RMCF0_	6/3545	Illegal restricted mode fault.	(D,C)
SVCF0_	6/3760	Illegal SVC.	(D,C)
WARMH_	4/1022	Can't warm start	(C)
Any other location.			(D,C)

ACTIONS TO TAKE AFTER HALT CONDITION IS DETERMINED

D -- Note down register set (if RSAVPTR.NE.0) and take a tape dump.
C -- Cold start.
W -- Warm start.
W* -- Warm start is possible only if a user page got the parity error.
M -- Map out bad page.

MEMORY PARITY ERRORS

On halt (at MEMPA_): X = user number, A = physical page number, B = word number. Hit start to automatically map out bad page.

c(MMAP+PPN) --> HMAP for page (≤ '6200 => supervisor page).

c(PTUSEG+(PHMAP.RS.6-'40)=user number

Manual mapout: MMAP+PPN<-- -1, HMAP <-- 0

MEMORY/REGISTER DISPLAY

1) Select FETCH Y on rotary switch (this stops the machine).
2) Select ADDRESS on ADDRESS/DATA toggle, hit DATA CLEAR, and depress address or register number in switches as shown below.
3) After switches depressed/set, select DATA and depress START. The lights now show contents of selected location on register.

For memory references, raised switches are in top row, dialed switches in bottom row. 'H/L' selects high (raised) or low (middle position) side of register.

```
                  1                12  16
300 REGS:         |0|000|000|000|0RR|RRR|
                                 REG #

                  1    5<--SEG NUM->16
MAPPED MEMORY:    |0|000|SSS|SSS|SSS|SSS|
                  |W|WWW|WWW|WWW|WWW|WWW|
                  <--WORD NUMBER------>

                  1   4           11   16
ABSOLUTE MEMORY:  |0|001|000|000|WWW|WWW|
                  |W|WWW|WWW|WWW|WWW|WWW|
                  <----WORD NUMBER---->

                  1   4              12  16
CURR. REG. SET:   |1|00H/L|000|000|0RR|RRR|
                                   REG #

                  1 2  4           11   16
ABSOLUTE REG.:    |1|10H/L|000|00R|RRR|RRR|
                                 REG #
```

Virtual addresses (but not absolute memory or registers) can be displayed while PRIMOS IV is running. Place ADDRESS/DATA toggle on DATA and enter segment number/word number as follows ('D' => depress switch):

```
                  1 2                  16
CLEAR SEGNO:      |D|D00|000|000|000|000|

                  1     5              16
DEPRESS SEGNO:    |D|000|DDD|DDD|DDD|DDD|
                        <---SEG NUM--->

                  1                    16
RAISE WRDNO:      |W|WWW|WWW|WWW|WWW|WWW|
                  <----WORD NUMBER---->
```

NOTE: If referenced page is not in memory, zeroes will be displayed (i.e., page faults are ignored).

MEMORY SCAN

1) Master clear, load '777 into the PC (location 7), select RUN, place ADDRESS/DATA switch on DATA, data switches in neutral position, hit START. (PRIMOS IV must be in memory.)
2) When the data lights change, a bad memory location has been found. The display with all switches neutral is the word number within the page. To display the physical page number, raise switch 15. To display the contents of the location, raise switch 14 (drop switch 15).
3) To continue the memory scan, type any character on the system console (ASR).
4) The scan will halt when the end of memory is reached. Hitting START will restart the scan.

TAPE DUMP

1) Mount non-write-protected tape on first magtape controller, drive 1. Ensure only one unit dialed to 1. Tape should be at load point and online. (PRIMOS IV or V must be in memory.)
2) Master clear, set PC (location 7) from '1000 to '776, select RUN, hit START.
3) When done, the tape dump program will rewind the tape and halt. Perform WARM or COLD START as appropriate.

WARM START

Master clear, select RUN, hit START twice.

NOTE: Warm start is possible only on CPUs with REV 1Ø microcode or later, otherwise an immediate halt at WARMH_ will occur.

7 PERIPHERAL I/O

ADDRESSES

00	Polling	40	PRIMAD (AIS)
01	Paper Tape Reader	41	Digital Input 1
02	Paper Tape Punch	42	Digital Input 2
03	Unit Record Controller 1	43	Digital Output 1
04	TTY	44	Digital Output 2
05	Unit Record Controller 2	45	Analog Output
06	Interproc. Channel (IPC)	46	Computer Prod. IF
07	--	47	CAMAC Interface
10	--	50	HSSMLC 1
11	--	51	HSSMLC 2
12	Diskette	52	AMLC 3
13	Magtape Controller 2	53	AMLC 2
14	Magtape Controller 1	54	AMLC 1
15	RIOX I/O Bus Switch	55	MACI Autocall
16	RIOX MPS	56	SMLC
17	--	57	--
20	Panel, Real Time Clock	60	Gen. Purp. IF Board
21	Disk (4002 Controller)	61	Ringnet Controller
22	Fixed Head Disk	62	GPIB
23	30 Megabyte Disk	63	GPIB
24	Writeable Control Store	64	GPIB
25	Moveable Head Disk	65	GPIB
26	Storage Module 1	66	GPIB
27	Storage Module 2	67	GPIB
30	IOC 1 (Parallel I/O)	70	GPIB Test
31	IOC 2	71	ADAGE GP/400 IF
32	--	72	--
33	VERSATEC	73	--
34	VERSATEC	74	--
35	AMLC 4	75	--
36	ELFBUS Controller 1	76	--
37	ELFBUS Controller 2	77	I/O Bus Test

AMLC

OTA 01 -- Set Line Configuration

1-4	170000	Line Number
5	004000	Unused
6	002000	Data Set Control
7	001000	Loop Line
8-10	000700	Line Speed: xxx0xx - 110 BAUD
		xxx1xx - 134.5
		xxx2xx - 300
		xxx3xx - 1200
		xxx4xx - Pgmed Clock
		xxx5xx - Pgmed Clock
		xxx6xx - Pgmed Clock
		xxx7xx - Pgmed Clock
11	000040	Unused
12	000020	0 => 1 Stop Bit, 1 => 2
13	000010	1 => Disable Parity
14	000004	0 => Odd Parity, 1 => Even
15-16	000003	Char Len: xxxxx0 - 5 Bits
		xxxxx1 - 7 Bits
		xxxxx2 - 6 Bits
		xxxxx3 - 8 Bits

OTA 02 -- Set Line Control

1-4	170000	Line Number
5-10	007700	Unused
11	000040	1 => Enable Char Time Interrupt
12	000020	Unused
13	000010	1 => Enable Transmit
14	000004	1 => Enable Echo Back
15	000002	1 => Receive Off, Report Break
16	000001	1 => Enable Receive

ASR

		SOC	SOC
BAUD	OPTION-A	CTL 1	CTL 2
110	110	27	740**
300	1010	76	340**
1200	2010	373	340**
9600	3410	3735	340**

** = number of delays used by BOOT, PRIMOS

DISK CONTROLLERS

Disk Channel Program Definitions

Mnem	Op Code	Execution Time (u-s)	Order	Fields	
DHLT	0	6	Halt		
SFORM	2		Format	Rec Size	13-16
				Track Addr	23-32
				# Records	33-40
				Head Addr	44-48
SSEEK	3	7.5	Seek	Restore	17
				Clear	18
				Track Addr	23-32
DSEL	4	7.5	Select	MHD	29-32
SREAD	5		Read	Rec Size	13-16
				Offset	17-20
				SR	21
				Track Addr	23-32
				Rec Addr	33-40
				Head Addr	44-48
SWRITE	6			Rec Size	13-16
				Track Addr	23-32
				Rec Addr	33-40
				Head Addr	44-48
DSTALL	7	210	Stall		
DSTAT	9	9	Input Status	Mem Addr	17-32
SSTOR	A	9	Store	Diag Addr	16
				Mem Addr	17-32
DOAR	B	9	Input OAR	Mem Addr	16
SLOAD	C	9	Load	Diag Addr	16
				Mem Addr	17-32
SDMA	D	6	Channel Address	Chain	13-16
				Chan Addr	17-32
DINT	E	6+CPU	Interrupt	Vect Addr	17-32
DTRAN	F	6	Transfer	Trans Addr	17-32

```
 1     4|5          10
|  op   |            | ...
| code  |    mask    |
```

bit 5 = 0, do not execute inst if:
bit 5 = 1, execute inst if:

set bit	
6	No function but reserved for "selected diskfile is write protected."
6	Last read or write record inst caused a DMA overrun, check error, controller parity error or header check failure (status word bits 2,4,5, or 6 set).
8	Selected MHD is seeking.
9	Selected diskfile has an error condition (status word bits 14, 15, or 16 are set).
10	For dual port operation only. Selected diskfile is busy servicing the "other" controller.

Disk Device Numbers (DVNO)

1-4	170000	(Offset to First Head)/2
5-8	007400	(Number of Heads)/2
9	000200	0=>Controller 1, 1=>Controller 2
10-13	000170	Type of Controller: xxx00x 4000 MHD xxx01x 4000 FHD xxx02x Diskette xxx03x 4003 8 Sectors/Track xxx04x 4003 FHD xxx05x 4003 32 Sectors/Track xxx06x 4004 Storage Module xxx07x-xxx17x Undefined
14-15	000006	Unit (Inc. bit 16 for Diskette)
16	000001	Diskette: Low Bit of Unit 4003 Controller: 0 => Top, 1 => Bottom Storage Module: LSB of Number Heads

Disk Errors

Option B (4000 Controller)

--	177777	Bad Record Identifier
--	177776	Device Not Ready
1	100000	Data Transfer Complete
2	040000	R/W Past End of Record
3	020000	Unused
4	010000	Stack Available
5	004000	Seek Complete OK
6	002000	Write Protect Violation
7	001000	Not Ready
8	000400	Command Error
9	000200	Checksum Error
10	000100	DMX Overrun
11	000040	Stack Overflow
12-15	000036	Unused
16	000001	Not Ready (Software)
--	000000	Redundant Int. (Warm Start)

Option B-Prime (4001 Controller)

--	177777	Bad Record Identifier
--	177776	Device Not Ready
--	177775	Memory Parity Error During DMX
1	100000	Bit 1 Always Set
2	040000	DMX Overrun
3	020000	Write Protect Status
4	010000	Checksum Error
5-9	007600	Unused
10	000100	Unit 1 Seeking
11	000040	Unit 2 Seeking
12	000020	Unit 3 Seeking
13	000010	Unit 4 Seeking
14	000004	Illegal Seek
15	000002	Malfunction Detected
16	000001	Not Ready (Software)
--	000000	Redundant Int. (Warm Start)

Diskette Controller

--	177777	Bad Record Identifier
--	177776	Device Not Ready
1	100000	Normal End of Instruction
2	040000	Sector Not Found
3	020000	Checksum Error on Sector ID
4	010000	Track Error (head misposition)
5	004000	Bad CTA or Not Ready
6	002000	Deleted Data Mark Read
7	001000	DMX Overrun
8	000400	Chksum err, Write Prot. Violation, Inoperable on Write or Format
9-15	000376	Unused
16	000001	Not Ready
--	000000	Redundant Int. (Warm Start)

Storage Module (4004 Controller)

--	177777	Bad Record Identifier
--	177776	Device Not Ready
--	177775	Memory Parity Error During DMX
1	100000	Bit 1 Always On
2	040000	DMA Overrun
3	020000	Write Protect
4	010000	Read Check
5	004000	Data Parity Error
6	002000	Header Check
7-10	001700	Unused
11	000040	Busy (Dual Port Only)
12	000020	Unused
13	000010	Seeking
14	000004	Illegal Seek
15	000002	Select Error
16	000001	Not Ready
--	000000	Redundant Int. (Warm Start)

Disk Sizes

*** Disk types and sizes ***
to be supplied at next update

DMX CONTROL WORDS

DMA

```
    1           12 13
 Ø | -WORD COUNT  ØØØØ|
 1 | STARTING ADDRESS |
```

```
              1    5 6          16
OTA '14dd: |N|NNN|Ø CHAN ADDRESS|
```

NNNN = Number of channels - 1.

DMC

```
    1           16
 Ø | START ADDRESS |
 1 |  END  ADDRESS |
```

```
              1  4 5 6          16
OTA '14dd: |N|NNN|1 CHAN ADDRESS|
```

NNNN = Number channels - 1.

DMQ

```
    1                         16
 Ø |T|TTT|TTT|TTT|TTT|TTT|    (Read Pointer)
 1 |B|BBB|BBB|BBB|BBB|BBB|    (Write pointer)
 2 |Ø|---|---|---|PPP|PPP|    (H. O. bits phy addr)
 3 |M|MMM|MMM|MMM|MMM|MMM|    (mask)
```

mask = len-1 of Q
len = 2**K, 4≤K≤1Ø
(Queue must be on 2**K boundary.)

INPUT: End of Range if no room.
OUTPUT: EOR if empty (not w/last entry).

DMT

Device Defined.

MAGTAPE

Command Bit Definitions

1	1ØØØØØ	Select Transport (bits 9-12)
2	Ø4ØØØØ	Ø=>File Operation, 1=>Record Op
3	Ø2ØØØØ	Ø=>Read/Write Op, 1=>Spacing Op
4	Ø1ØØØØ	1=>9-Track Read and Correct
5	ØØ4ØØØ	Ø=>Binary, 1=>BCD (7-track only)
6	ØØ2ØØØ	Ø=>7-Track Transport, 1=>9-Track
7	ØØ1ØØØ	Unused
8	ØØØ4ØØ	1=>2 Characters per Word
9	ØØØ2ØØ	1=>Forward Motion (bits 1Ø,11=Ø)
1Ø	ØØØ1ØØ	1=>Reverse Motion (bits 9,11,12=Ø)
11	ØØØØ4Ø	1=>Rewind (bits 9,1Ø,12=Ø)
12	ØØØØ2Ø	1=>Write Order
13	ØØØØ1Ø	Select Transport Ø
14	ØØØØØ4	Select Transport 1
15	ØØØØØ2	Select Transport 2
16	ØØØØØ1	Select Transport 3

Magtape Commands

1ØØØØØ	Select Transport (7 and 9 track)
ØØØØ4Ø	Rewind to BOT (7 and 9 track)
Ø221ØØ	Backspace File Mark, 9-track
Ø2Ø1ØØ	Backspace File Mark, 7-track
Ø621ØØ	Backspace Record, 9-track
Ø6Ø1ØØ	Backspace Record, 7-track
Ø2222Ø	Write File Mark, 9-track
Ø2Ø22Ø	Write File Mark, 7-track
Ø622ØØ	Forward Space Record, 9-track
Ø6Ø2ØØ	Forward Space Record, 7-track
Ø222ØØ	Forward Space File Mark, 9-track
Ø2Ø2ØØ	Forward Space File Mark, 7-track
Ø4222Ø	Write Record One Char/Word, 9-track
Ø4262Ø	Write Record Two Char/Word, 9-track
Ø422ØØ	Read Record One Char/Word, 9-track
Ø426ØØ	Read Record Two Char/Word, 9-track
Ø522ØØ	Read/Correct Record One Char/Word, 9-track
Ø526ØØ	Read/Correct Record Two Char/Word, 9-track
Ø4Ø22Ø	Write Binary Record One Char/Word, 7-track
Ø4Ø62Ø	Write Binary Record Two Char/Word, 7-track
Ø4422Ø	Write BCD Record One Char/Word, 7-track
Ø4462Ø	Write BCD Record Two Char/Word, 7-track
Ø4Ø2ØØ	Read Binary Record One Char/Word, 7-track
Ø4Ø6ØØ	Read Binary Record Two Char/Word, 7-track
Ø442ØØ	Read BCD Record One Char/Word, 7-track
Ø446ØØ	Read BCD Record Two Char/Word, 7-track

Magtape Status

1	1ØØØØØ	Parity Error
2	Ø4ØØØØ	Runaway Tape
3	Ø2ØØØØ	CRC Error
4	Ø1ØØØØ	LRC Error
5	ØØ4ØØØ	Low DMX Range
6	ØØ2ØØØ	Permanent Error
7	ØØ1ØØØ	Read-After-Write (RAW) Error
8	ØØØ4ØØ	File Mark Detected
9	ØØØ2ØØ	Ready
1Ø	ØØØ1ØØ	Online
11	ØØØØ4Ø	End of Tape Detected
12	ØØØØ2Ø	Rewinding
13	ØØØØ1Ø	Beginning of Tape (at Load Point)
14	ØØØØØ4	Tape is Write-Protected
15	ØØØØØ2	DMX Overrun
16	ØØØØØ1	Rewind Complete

(ØØØ3ØØ or ØØØ3Ø4 - Normal Completion)

PROGRAMMED I/O (PIO)

OCP -- Output Control Pulse

Ø3FFDD FF=Function, DD=Device Address

SKS -- Skip on Condition

13CCDD CC=Condition, DD=Device Address

INA -- Input to A-Register

Ø7FFDD FF=Function, DD=Device Address

No skip for device '2Ø
Always skips if status register input.

OTA -- Output from A=Register

17FFDD FF=Function, DD=Device Address

No skip if device '2Ø.

Standard Functions

FF	OCP	SKS	INA	OTA
ØØ		Ready	Data Reg	
Ø1		Not Busy		
Ø2				
Ø3				
Ø4		Not Interrupting		
Ø5				
Ø6				
Ø7				
1Ø				
11			Input ID	
12	Normal Mode			
13	Diagnostic Mode			
14	Ack Interrupt			DMX Channel
15	Set Int Mask			
16	Reset Int Mask			Int Vect Addr
17	Initialize			

8 PRIMOS IV

ABORT FLAGS

PCB+4, ABSAVE at 6000/10

100000	MINALM	ONE MINUTE ABORT FLAG
040000	SMLALM	SLMC ALARM
020000	NETALM	NETWORK ALARM
004000	WRMALM	WARM START ALARM
000200	MT1ALM	MTDONE, CONTROLLER 1
000100	MT2ALM	MTDONE, CONTROLLER 2
000020	LOGALM	FORCED LOGOUT ALARM
000010	DISALM	DISCONNECT ALARM
000004	TMOALM	TIMEOUT ALARM
000002	QUTALM	QUIT ALARM
000001	TSEALM	TIME SLICE END (FIRMWARE)

COMMONS

*** List of COMMONS ***
to be supplied at next update

ERRVEC

In PUDCOM at 6000/106

ERRVEC(1) ALTVAL
ERRVEC(2) ALTVAL(2)
ERRVEC(3) NAME (0=>NO NAME)
ERRVEC(4) "
ERRVEC(5) "
ERRVEC(6) "
ERRVEC(7) WORD NUMBER OF MSG OR 0
ERRVEC(8) LENGTH (IN CHARS) OF MSG
ERRVEC(9) SEGMENT NUMBER OF MSG

V-mode error messages saved in ERRVEC:

MESSAGE	ERRVEC(1-2)
ACCESS VIOLATION	PB of instr causing violation
ILLEGAL PAGE REF	32-bit ptr into ill page
ILLEGAL SEGNO	32-bit ptr into ill seg
POINTER FAULT	PB of instr causing fault
NO AVAIL SEGMENTS	32-bit ptr to seg that system attempted to create
UNDEFINED GATE	32-bit ptr into gate seg

FIGCOM

Starts at 4/700

LOC	NAME	DFLT	DEFINITION
700	LOUTQM	1000	INACTIVE MINUTES TO AUTO LOGOUT
701	RWLOCK	1	SYSTEM READ/WRITE LOCK: 0 - 1 READER OR 1 WRITER 1 - N READERS OR 1 WRITER 3 - N READERS AND 1 WRITER 5 - N READERS AND N WRITERS
702	DONSTP	0	0=>LOGOUT PHANTOMS ON WARM START 1=>CONTINUE PHANTOMS ON WARM START
703	DLOGOT	0	0=>IGNORE DISCONNECT 1=>FORCE LOGOUT ON DISCONNECT
704	DEFERA	242	DEFAULT ERASE CHARACTER = "
705	DEFKIL	277	DEFAULT KILL CHARACTER = ?
706	PRI500		1=>P500
707	VERSIO		PRIMOS REVISION ID (ASCII)
720	NLGPRT		1=>INHIBIT LOGIN MESSAGES
721	LOGOUT		1=>CAN'T LOGIN WHILE LOGGED IN
722	LRQUOT	10000	LOGREC QUOTA

INTERNAL CALLING SEQUENCES

```
AINIT  VERSIO,MEMSIZ
ALCONF A,B
ALONF  BNO
AMINIT --
ASRDIM --
BADDSK DISK  (LOG FCN)
BFDEQU BUFCON,NW (FCN)
BFENQU BUFCON,NW
BFGETR BUFCON,NW (FCN)
BFRELS BUFCON
BOOT   --
BRPDIM --
BRPONF FLAG
BRPOTA CHAR     (LOG FCN)
BUFCHK BNO,NUMCHARS    (LOG FCN)
BUFCLR BNO
CE2DIM --
CENDIM --
CGETBK SIZE,ADDR,COND,ALTRTN
CLNLUN USRBLK,LUNADR,STAT
CLOSE$ UNIT
CNFLCT KEY,PDEV  (LOG FCN)
COMXIT --
COPYUP --
CRDONF FLAG
DATE$  XXX    (FCN)
DELAY  A,B,C,ALTRTN
DELBKQ QNM,BKN,ALTRTN
DEMOTE --
DEQUE  NUM,ADDR,COUNT,ALTRTN
```

```
DEVCHK USR    (FCN)
DEVONF DEVNDX,FLAG
DISMSG USRBLK,LUNADR
DOSSUB --
DSKEQV DISKA,DISKB     (LOG FCN)
DUMF   FLAG
ENQUEB NUM,ADDR,COUNT
ENQUET NUM,ADDR,COUNT
ERRST$ LCODE,KEY,TEXT,TXTLEN,MYNAME,MYLEN
EXPNAM INNAME,NAMLEN,OUTNAM
F$HT   CODE
FAMERR --
FAMMSG FUNNO,FAMNO,RA,LOC32(A1),N1,LOC32(A2),N2,
       LOC32(A3),N3,LOC32(A4),N4,LOC32(A5),N5,
       LOC32(A6),N6
FMLIOB BNO,CHAR    (LOG FCN)
FNDLUN USRBLK,LSRCID,LUNADR,ALTRTN
FREEBL --
GCHAR  LOC32(ARRAY),CHARPTR
GET    LOC32(WORD)     (FCN)
GETBLK SIZE,ADDR,ALTRTN
GETCNT SIZE,COUNT
GETLUN USRBLK,SW,LUNADR,ALTRTN
GETRBK ALTRTN
GETREC RA32,DVNO,CODE   (INT*4 FCN)
GETREG TVEC
GETSBK BLKAD
GETSEG USR,SEGNO
GETTBK BLKAD,ALTRTN
GETUBK USRBLK,ALTRTN
GROSS  --
GTSTAT XXX     (FCN)
HCRONF FLAG
HDFILL TGTN,SRCN,TGTU,MSGBLK,LINTAB,NETHDR,FLGW
HPRONØ FLAG
HPRON1 FLAG
INETMN --
ININET --
INUSRC --
IPCPTM LINEN
ISPREM A,B,C   (FCN)
ITLBNZ --
LGET   LOC32(WORD)     (INT*4 FCN)
LINTST LINDFN,ALTRTN
LISTF  --
LOCATE KEY,RA32,LDEV
LOCK   KEY,SEMAPHORE     (LOG FCN)
LOCKFS --          (SHORT CALL)
LOCKPG USR,KEY,PTR32,NW
LOGEV1 MSG
LOGEV2 --
LOGIN  KEY,UNAME,NAMLEN,USNAME,LDEV
LSTORE LOC32(WORD),DATA
MAPIO  LOC32(USRBFR),NW,PAGE-MAP-ENTRY   (INT*4 FCN)
MAPNDX USR,SEGNO (FCN)
MISSIN ARG   (LOG FCN)
MISZER ARG   (FCN)
```

```
MODTS  DELTA
MOV32P LOC32(FROM),LOC32(TO),NW
MOVNAM INAME,ILEN,ONAME,OLEN,TRULEN
MOVS2S FRMSEG,LOC16(FROM),TOSEG,LOC16(TO),NW
MOVUTU FRMUSR,FRMSEG,LOC16(FROM),TOUSR,TOSEG,
       LOC16(TO),NW
MPINIT --
MSGOUT KEY,USR,LOC32(BFR)  (FCN)
MTDONE CTRLR-INDEX
NETALM --
NETMAN --
NETMSG FLGW,LUNADR
NETXEC --
NEWDAM DRWP,UNIT,NRAL,CODE
NOTIFY OPTION,SEMAPHORE
OERRTN ALTVAL,ALTRTN,CODE,TEXT,TXTLEN,NAME,NAMLEN
PABORT ABORT-FLAGS
PAGTUR LOC32(VIRTADR)
PHINIT --
PNDNAM SYSN,NETABL,ALTRTN
PRCFP  ABORT-FLAGS
PRIPC  --
PRSMLC --
PRWERR ALTVAL,CODEV
PTRAP  INSTRUCTION
PTRDIM --
PTRINA CHAR    (LOG FCN)
PTRONF FLAG
QCHECK NUM,ELQUED
QTRVRS NUM,BLKN,ADDR,COUNT,ALTRTN
QUITON --              (SHORT CALL)
RMANLZ INST,FT,OP,ACT,DEV
ROUTER RTNGBK,HDROFF,MSGBLK,COUNT
RTNBLK ADDR
RTNREC RA32,DVNO
RTNSEG SEGNO
RUNUSR FAMBLK,NW,ERVEC
SCHAR  LOC32(ARRAY),CPTR,CHAR
SCHED  RESET-VAL [,QUEUE-SEMAPHORE]
SDWNDX USR,SEGNO    (FCN)
SEEK   CYL,DVNO
SEMTN  SEMAPHORE,INT1,INT2,CODE
SEND   USR,LUNADR,ALTRTN
SETABT KEY,USR,ALARM
SETREG TVEC,PARFLG
SHUTDN --
SLABRT LPN
SLCLDB --
SLCONF FLAG
SLERF  KEY
SMLCEX SVCF,LPN
SRWREC KEY,PBAV,NWV,NCH,RA32,DVNO,ALTRTN
STAC   NEWVAL,OLDVAL,VAR     (LOG FCN)
STIMER TENTHS
STORE  LOC32(WORD),DATA
SYSERR --
T1OU   USR,CHAR
```

```
TEXTO$ NAME,NAMLEN,TRULEN,TEXTOK
TEXTOK FNAMEBFR
TIME   USR,TIM
TIME$  XXX     (FCN)
TODEC  USR,NUM,LEAD-CHAR/FIELDWIDTH
TOLIOB BNO,CHAR
TPIOS  KEY,PBA,PN,RA32,ALTRTN
TRNMSG LINTAB,MSGBLK,CNT,MSGVEC
TRUNC$ UNIT,CODE
TRWRAT KEY,LDEV
UMAPIO LOC32(USRBFR),NW
UNLKFS --      (SHORT CALL)
UNLOCK KEY,SEMAPHORE
UPDATE KEY,RA32,LDEV
UPUSR  LOC32(USRADDR),DATA,NWD
VALID  TNAME,NETTAB,ALTRTN,NODE
VGONF  FLAG
VGINIT --
WAIT   SEMAPHORE
WREC   PBA,NW,NCH,CRA32,DVNO,ALTRTN
XEQUSR --
************************************************
```

LOCKS, LCKCOM

Locks are semaphores used to control access to serially reusable resources. Located in LCKCOM (SEG 4), source file N1LOCK.

*** Addition LOCK Information ***
to be supplied at next update

MMAP (MEMORY MAP)

One entry per physical page of memory.

MMAP + n:

```
< Ø => Page 'n' unavailable
= Ø => Page 'n' available
> Ø => In use, --> HMAP entry for page
```

MAXPAG = number pages memory in use.

PAGE MAPS

See under Section on CPU.

PTUSEG

PTUSEG(2,KSEG) (SEG 4)

```
PTUSEG(1,N)  Owner of Page Map N
PTUSEG(2,N)  Segment Number for Page Map N
```

PUDCOM

```
ØØ6ØØØ  SUPSTK    SN FOR SUPERVISOR STACK
ØØ2ØØØ  STKSIZ    ONE PAGE OF STACK

ØØØØØØ  PUDCOM SUPFRE(2),SUPEXT(2)

ØØØØØ4  CUSR      1  CURRENT USER NUMBER
ØØØØØ5  LUSR      1  USERCOM INDEX
ØØØØØ6  VRTSSW    1  VIRTUAL SENSE SWITCHES
ØØØØØ7  INHPRF    1  INHIBIT-PROCESS-FAULT COUNTER
ØØØØ1Ø  ABSAVE    1  SAVED ABORT FLAGS
ØØØØ11  HILOCK    1  ADR HIGHEST OWNED ORDERED LOCK
ØØØØ11  LCKOWN
ØØØØ12  OWNFS     1  NUMBER ORDERED LOCKS OWNED
ØØØØ13  QUITF     1  QUIT FLAG, INHIBIT COUNT
ØØØØ14  ASRCWD    1  ASR CONTROLS
ØØØØ15  ERASCH    1  CHARACTER ERASE CHARACTER
ØØØØ16  KILLCH    1  LINE DELETE CHARACTER
ØØØØ17  TKNSAV    1  RDTK$$ SAVEAREA
ØØØØ2Ø  COMPAR    4Ø COMMAND LINE INPUT BUFFER
ØØØØ7Ø  COMSWI    1  COMMAND INPUT SWITCH
ØØØØ71  COMUNI    1  COMMAND INPUT UNIT
ØØØØ72  COUSWI    1  COMMAND OUTPUT SWITCH
ØØØØ73  COUPTR    1  COULIN CHARACTER POINTER
ØØØØ74  COULIN    1Ø COMMAND OUTPUT BUFFER
ØØØ1Ø6  ERRVEC    9  ERRVEC
ØØØ117  XSAVE     1  TEMP SAVE FOR X IN FAULT RTES
ØØØ12Ø  RVSA      1  START ADDRESS IN RVEC
ØØØ121  RVEA      1  END ADDRESS IN RVEC
ØØØ122  HMAPPP
ØØØ124  BUFNEW
ØØØ125  USRETM
ØØØ126  SUPSF     START OF FIRST RING Ø STACK FRAME
ØØØ13Ø  RVPB      2  PBH AND PBL
ØØØ136  RVKEYS    1  KEYS
ØØØ137  RVPBCL    1  LOC(PCL)+2 ON RING Ø PCL ENTRY
ØØØ143  RVEC      29 REGISTER SAVEAREA
ØØØ143  RVMASK    1  SAVE MASK IN RVEC
ØØØ172  PUDEND    END OF PUDCOM
```

SEGMENT USAGE BY PRIMOS

SEGMENT	CONTENTS
Ø	LOC '61 (OPTION-A memory increment cell) DMC channels for AMLC, SMLC, MAG TAPE AMLC buffers DISK driver (DVDISK) Disk I/O windows (4 pages) Mag tape I/O windows (6 pages) Mag tape dump window (1 page) IPC I/O window (2 pages) SMLC I/O windows (12 pages)
1	Associative bfrs for file system (64 pages)
2,3	MOVU2U segment windows
4	Interrupt catchers (phantoms) Check catchers Semaphores Ready PCB list (loc '6ØØ) Configuration common (FIGCOM) (loc '7ØØ) Crash 9 trk magtape dump program (loc '776) Memory parity scanner (loc '777) WARM restart routine (loc '1ØØØ) COLD start routine (loc '14ØØ) Memory usage map (MMAP) Page maps (HMAP, LMAP) Segment tables Process control blocks (PCBs) Interrupt fault table Interrupt stack
5	Gate segment for direct-entrance PCLs
6	TMAIN, including: Supervisor and user fault catchers SVC front-ends Supervisor locked data (SUPCOM) Clock process Kernel procedures
7	User terminal buffers
1Ø	Per-user unlocked data (USRCOM)
11	File system procedures
12	Network data and procedures SMLC data and procedures
6ØØØ	Ring Ø stack segment (one per user)

SEMAPHORES (SEMCOM)

*** Refer to SEMAPHORES in Section II ***

*** Addition SEMAPHORE Information ***
to be supplied in next update

SVC INTERLUDE

```
ENTRY DAC    **
      SVC
      OCT    CODE
  1   100000 1 => interlude call
  2   040000 1 => bounce
3-4   030000 Unused
5-16  00**** SVC number
```

USRCOM

```
SEG 10/(USRNO-1)*'505 ...

+1   UNITAB:        0 ≤ UNIT ≤ '21
        +UNIT*17 +0  VSTAT
                 +1  VBRA(2)
                 +3  VDVNO
                 +4  VDCRA(2)
                 +6  VDRWP
                 +10 VCRA(2)
                 +12 VRWP
                 +13 VPRIV
                 +14 VPOPRA(2)
                 +16 VPOPRW
+417 CUFD:
     +0  CURRENT UFD NAME
     +20 CFDBRA(2)
     +22 CFDDEV
     +23 CFDPOP(2)
     +25 CFDOWN
     +26 CFDLEN
     +27 CFDPRA(2)
     +31 CFDPRW

+451 HOMUFD:
     +0  HOME UFD NAME
     +20 HOMBRA(2)
     +22 HOMDEV
     +23 HOMPOP(2)
     +25 HOMOWN
     +26 HOMLEN
     +27 HOMPRA(2)
     +31 HOMPRW

+503 LOGNAM(3)
```

VQUTM

1	1 MIN.	1 MINUTE (UPDATE, LOGEV2, ETC.)
2	16.5 MSEC	PUNCH DIM
3	--	DIGITAL INPUT
4	112 MSEC	ASR DIM
5	100 MSEC	TENTH SECOND (STIMER QUEUES)
6	--	UNUSED
7	--	UNUSED
8	--	UNUSED
9	--	UNUSED
10	1/2 SEC	SMLCEX ALARM
11	10 SEC	NETWORK GROSS TIMER
12	1 SEC	IPC PROTOCOL TIMER
13	1/2 SEC	REMOTE USER POLL

9 SVC INFORMATION

SVC CALLING SEQUENCES

* => Also Direct Entrance Call.

ATCH$$ -- Attach to UFD

```
CALL ATCH$$ (UFDNAM, NAMLEN, LDISK, PASSWD,
             KEY, CODE)                          (*15ØØ)

CALL ATTAC$ (UFDNAM, NAMLEN, LDISK, PASSWD,
             KEY, LOC(CODE))                     (14ØØ)

CALL ATTACH (UFDNAM, LDISK, PASSWD, KEY,
             ALTRTN)                             (Ø1ØØ)

                  ****** KEY    ******
K$IMFD = :Ø              UFD IS IN MFD
K$ICUR = :2              UFD IS IN CURRENT UFD
                  ****** KEYMOD ******
K$SETC = :Ø              SET CURRENT UFD (DO NOT SET HOME)
K$SETH = :1              SET HOME UFD (AS WELL AS CURRENT)
                  ****** UFDNAM ******
K$HOME = :Ø              RETURN TO HOME UFD (KEY=K$IMFD)
                  ****** LDISK  ******
K$ALLD = :1ØØØØØ         SEARCH ALL DISKS
K$CURR = :177777         SEARCH MFD OF CURRENT DISK

                  ****** CODES  ******
                  E$NATT        E$MTUD
                  E$FMTF        E$DISK
                  E$PTRM        E$BPAS
                  E$BKEY        E$BUFD
```

BREAK$ -- Disable/Enable Quits

```
CALL BREAK$ (KEY)                                (*Ø5Ø7)

                  ****** KEY    ******
                    = :Ø    INHIBITS QUITS
                    = :1    ENABLE QUITS
```


C1IN -- Read Character from Command Stream

```
CALL C1IN (CHAR)                                 (*Ø6Ø1)
```

CMREAD -- Read Last Command Line

```
CALL CMREAD (BUFF(18))                           (Ø6Ø2)

BUFF(1-3)    First name or blanks
BUFF(4-6)    Second name or blanks
BUFF(7-9)    Third name or blanks
BUFF(1Ø)     First octal parameter or Ø
   ...
BUFF(18)     Ninth octal parameter or Ø
```

CNAM$$ -- Change a Filename

```
CALL CNAM$$ (OLDNAM, OLDLEN, NEWNAM, NEWLEN,
             CODE)                               (*1515)

CALL CNAME$ (OLDNAM, OLDLEN, NEWNAM, NEWLEN,
             LOC(CODE))                          (1415)

CALL CNAME (OLDNAM, NEWNAM, ALTRTN)              (Ø113)

             ****** CODES  ******
             E$BNAM        E$FNTF
             E$BUFD        E$IREM
             E$EOF         E$NRIT
             E$EXST
```

CNIN$ -- Raw Data Input from Command Stream

```
CALL CNIN$ (BUFF, CHARCNT, STATV(3))             (*Ø6Ø4)
```

COMANL -- Read Command Line

```
CALL COMANL                                      (*Ø6ØØ)
```

COMI$$ -- Switch Command Input Stream

```
CALL COMI$$ (FILNAM, NAMLEN, UNIT, CODE)          (*1516)

CALL COMIN$ (FILNAM, NAMLEN, UNIT, LOC(CODE)) (1416)

CALL COMINP (FILNAM, UNIT, ALTRTN)                (0603)
```

```
****** CODES   ******
E$BDAM         E$NATT
E$BUFD         E$NRIT
E$DIRE         E$PTRM
E$FIUS         E$UIUS
E$FNTF         E$UNOP
```

COMO$$ -- Control Routing of Terminal Output

```
CALL COMO$$ (KEY, FILNAM, NAMLEN, 0, CODE)     (*1523)
```

****** KEY	******
:1	TURN TTY OUTPUT OFF
:2	TURN TTY OUTPUT ON
:4	(RESERVED)
:10	TURN FILE OUTPUT OFF
:20	TURN FILE OUTPUT ON
:40	APPEND IF TURNING ON, CLOSE IF TURNING OFF
:100	TRUNCATE IF TURNING ON

```
****** CODES   ******
       E$EOF
```

CONECT -- Connect Logical Unit Number

```
CALL CONECT (TGTNAM, TGTUSR, LUN, DATA, STATV,
             LINTYP)                               (0401)
```


CREA$$ -- Create New UFD in Current UFD

```
CALL CREA$$ (UFDNAM, NAMLEN, OPASS, NPASS,
             CODE)                                 (*1501)

CALL CREAT$ (UFDNAM, NAMLEN, OPASS, NPASS,
             LOC(CODE))                            (1401)
```

```
****** CODES   ******
E$NATT         E$DKFL
E$BNAM         E$EXST
E$DISK         E$PTRM
E$BUFD
E$UIUS (bounce package only)
E$FIUS (bounce package only)
E$FDFL (old partition only)
```

D$INIT -- Initialize Disk Devices

```
CALL D$INIT (PDEV)                                 (0506)
```

DISCON -- Disconnect Logical Unit Number

```
CALL DISCON (LUN, DATA, STATV)                     (0410)
```

DUPLX$ -- Set/Return Terminal Characteristics

```
Integer = DUPLX$ (LWORD)
```

LWORD Contents:

bit		Meaning when on
1	100000	Half duplex
2	040000	No LF after CR
3	020000	XOFF/XON Recognition
4	010000	XOFF Received
5-8	007400	Reserved
9-10	000377	user number

LWORD = -1 => no update of LWORD.
Integer set to new LWORD setting.

ERKL$$ -- Read/Set Kill and Erase Character

```
CALL ERKL$$ (KEY, ERASEC, KILLC, CODE)          (*1524)

                ****** KEYS    ******
                    see COMO$$

                ****** CODES   ******
                      E$BKEY
                      E$BPAR
```

ERRPR$ -- Print Standard System Error Messages

```
CALL ERRPR$ (KEY, CODE, TEXT, TXTLEN, NAME,
             NAMLEN)                             (*1402)

                ****** KEY     ******
        K$NRTN = :0   NEVER RETURN TO USER
        K$SRTN = :1   RETURN AFTER START COMMAND
        K$IRTN = :2   IMMEDIATE RETURN TO USER

                ****** CODES   ******
                       E$EOF
                       E$LAST
```

ERRRTN -- Return Error Code

```
CALL ERRRTN (ALTRTN, NAME, MSG, MSGLEN)         (0106)
```

ERRSET -- Handle Error Messages

```
CALL ERRSET (ALTVAL, ALTRTN, NAME, MSG,
             MSGLEN)                            (0114)
```

EXIT -- Return to PRIMOS Command Level

```
CALL EXIT                                       (*0105)
```

FAMSVC -- (Called by FAM only)

```
CALL FAMSVC (A1, A2, A3, A4, A5, A6, ALTRTN)  (0400)
```


FORCEW -- Update Runit to Disk

```
CALL FORCEW (KEY, UNIT)                            (*0115)
```

GETCON -- Get Pending Connect Information

```
CALL GETCON (TARGET, USER, DATA, STATV)           (0402)
```

GETERR -- Get Error Messages

```
CALL GETERR (BUFF, NW)                             (0110)
```

GETERR is used after a return from PRWFIL:

On an alternate return:		On a normal return:	
ERRVEC(1)	Error code	PRWFIL:	
		ERRVEC(3)	Record number
		ERRVEC(4)	Word number
ERRVEC(2)	Alternate value	Key of read/write convenient:	
		ERRVEC(2)	No. of words transferred
		SEARCH:	
		ERRVEC(2)	File type

GINFO -- Return Operating System Information

```
CALL GINFO (BUFF, NW)
```

Return infomation for PRIMOS II:

BUFF(1)	Low bound of PRIMOS II and buffers (77777 octal if 64K PRIMOS II).
2	High bound of PRIMOS II (77777 octal if 64K PRIMOS II).
3	(not valid)
4	(not valid)
5	Low bound of PRIMOS II and buffer (64K PRIMOS II only).
6	High bound of 64K PRIMOS II.

Returned information for PrIMOS III, IV, and V:

xervec word	content
1	0
2	0
3-6	(not valid)

GPAS$$ -- Obtain UFD Passwords

```
CALL GPAS$$ (UFDNAM, NAMLEN, OPASS, NPASS,
             CODE)                                   (*1504)

CALL GPASS$ (UFDNAM, NAMLEN, OPASS, NPASS,
             CODE)                                   (1404)

             ****** CODES   ******
             E$NATT         E$NTUD
             E$FNTF         E$DISK
             E$PTRM         E$BUFD
             E$UIUS (bounce package only)
             E$FIUS (bounce package only)
```

NETLNK --

```
CALL NETLNK (STATV)                                  (0412)
```

NETWAT -- Put User to Sleep

```
CALL NETWAT                                          (0406)
```

NTSTAT -- Get Network Status

```
CALL NTSTAT (KEY, Pl, P2, ARRAY)                     (0407)
```

PRERR -- Prints Error Messages

```
CALL PRERR                                           (0111)
```


PRWF$$ -- Read-Write-Position SAM/DAM File

```
CALL PRWF$$ (KEY, FUNIT, LOC(BUFF), BUFLEN,
             POS32, RNW, CODE)                       (*1506)

CALL PRWFL$ (KEY, UNIT, LOC(BUFF), NW, POS,
             RNW, LOC(CODE))                         (1406)

CALL PRWFIL (KEY, UNIT, LOC (BUFF), NW, POS,
             ALTRTN)                                 (0300)

     KEY = RWKEY + POSKEY + MODE

                    ****** RWKEY   ******
K$READ = :1                READ
K$WRIT = :2                WRITE
K$POSN = :3                POSITION ONLY
K$TRNC = :4                TRUNCATE
K$RPOS = :5                READ CURRENT POSITION
                    ****** POSKEY ******
K$PRER = :0                PRE-POSITION RELATIVE
K$PREA = :10               PRE-POSITION ABSOLUTE
K$POSR = :20               POST-POSITION RELATIVE
K$POSA = :30               POST-POSITION ABSOLUTE
                    ****** MODE    ******
K$CONV = :400              CONVENIENT NUMBER OF WORDS

                    ****** CODES   ******
                    E$EOF          E$NOF
                    E$UNOP         E$DKFL
                    E$DISK         E$PTRM
                    E$BUNT         E$BOF
                    E$IREM         E$NTUD
                    E$NTSD
```

RDEN$$ -- Read UFD Entry

```
CALL RDEN$$ (KEY, FUNIT, BUFF, BUFLEN, RNW,
             NAM32, NAMLN, CODE)                (*1507)

CALL RDENT$ (KEY, UNIT, BUFF, BUFLEN, RNW,
             NAME32, NAMELEN, LOC(CODE))        (1407)
```

Entry Format:

```
 0  | ECW      |
 1  |F         |
    |  I       |
    |    L     |
    |      E   |
    |  ...     |  FILENAME (BLANK PADDED)
    |N         |
    |  A       |
    |    M     |
    |______E___|
17  | PROTEC   |  PROTECTION (OWNER/NONOWNER)
18  |RESERVED  |  RESERVED FOR FUTURE USE
19  | FILTYP   |  FILETYPE <--- END OF ENTRY FOR TYPE=1
20  | DATMOD   |  DATE LAST MODIFIED
21  | TIMMOD   |  TIME LAST MODIFIED
22  |RESERVED  |  RESERVED FOR FUTURE USE
23  |RESERVED  |  RESERVED FOR FUTURE USE
```

```
DATMOD = YYYYYYYMMMMDDDDD where:
         YYYYYYY is the year module 100
         MMMM is the month
         DDDDD is the day

DATMOD is held in binary seconds-since-midnight
       divided by 4.
```

```
                ****** KEY    ******
K$READ = :1            READ NEXT ENTRY
K$RSUB = :2            READ NEXT SUB-ENTRY
K$GPOS = :3            RETURN CURRENT POSITION IN UFD
K$UPOS = :4            POSITION IN UFD
K$NAME = :5            READ ENTRY SPECIFIED BY NAME
```

```
****** CODES  ******
E$EOF         E$NOF
E$UNOP        E$DISK
E$PTRM        E$BKEY
E$BUNT        E$BUFD
E$BFTS
```


RDLIN$ -- Read Line of Characters from ASCII File

```
CALL RDLIN$ (UNIT, LINE, NW, CODE)              (*1525)

CALL RDLIN  (UNIT, LINE, NW, ALTRTN)            (0202)
```

RDTK$$ -- Read Token from Command Line

```
CALL RDTK$$ (KEY, INFO(8), BUFF, BUFLEN,
             CODE)                              (*1517)

CALL RDTKN$ (KEY, INFO(8), BUFF, BUFLEN,
             LOC(CODE))                         (1417)
```

```
INFO(1):
     1  normal token
     2  register setting parameter
     5  null token
     6  end of line

INFO(2):  len in chars of token; null = 0 len

INFO(3):  further info about token --
     bit 1 (:100000)  dec conversion successful,
                      value returned in INFO(4).
     bit 2 (:040000)  oct conversion successful,
                      value returned in INFO(5).
                      This bit always set when
                      token type is 2.
     bit 3 (:020000)  token begins with unquoted
                      minus sign.
     bit 4 (:010000)  explicit postion for register
                      setting given, value returned
                      in INFO(4).
     bits 5 - 16:     reserved for future use.

INFO(4):  depend on flags set in INFO(3)

INFO(5):  depend on flags in INFO(3)

INFO(6) - INFO(8):  reserved for future use
```

```
****** CODES  ******
       E$BKEY
       E$BPAR
       E$BFTS
```

RECEIV -- Receive Message from Remote System

```
   CALL RECEIV (LUN, LOC(BUFF), NW, STATV)         (Ø4Ø4)
```

RECYCL -- Cycle to Next User

```
   CALL RECYCL                                     (*Ø5Ø5)
```

REST$$ -- Restore Memory Image from File

```
   CALL REST$$ (RVEC, NAME, NAMLEN, CODE)          (*152Ø)

   CALL RESTO$ (RVEC, NAME, NAMLEN, LOC(CODE))     (142Ø)

   CALL RESTOR (RVEC, NAME, ALTRTN)                (Ø1Ø3)

                     ****** CODES  ******
                        see SRCH$$
```

RESU$$ -- Resume Memory Image from File

```
   CALL RESU$$ (NAME, NAMLEN)                      (*1521)

   CALL RESUM$ (NAME, NAMLEN)                      (142Ø)

   CALL RESUME (NAME)                              (Ø1Ø4)

                     ****** CODES  ******
                        see SRCH$$
```

RJCON -- Reject Pending Connect

```
   CALL RJCON (TARGET, USER, STATV, NUMTYP)        (Ø4Ø3)
```

RREC -- Read Record From Disk to Memory

```
   CALL RREC (LOC(BUFF), BUFLEN, N, RA, PDEV,
              ALTRTN)                              (Ø5ØØ)

   CALL RRECL (LOC(BUFF), BUFLEN, N, RA32, PDEV,
               ALTRTN)                             (Ø516)
```

SATR$$ -- Set Attributes in UFD Entry

```
   CALL SATR$$ (KEY, NAME, NAMLEN, ARRAY, CODE)    (*151Ø)
                                                                 5Ø6)
   CALL SATTR$ (KEY, NAME, NAMLEN, ARRAY,
                LOC(CODE))                         (141Ø)
                                                                 Ø6)

                ****** KEY    ******
   K$PROT = :1         SET PROTECTION                            ØØ)
   K$DTIM = :2         SET DATE/TIME MODIFIED
   K$DMPB = :3         SET DUMPED BIT
   K$RWLK = :4         SET PER FILE READ/WRITE LOCK

                ****** CODES  ******
                E$NATT        E$FNTF
                E$DISK        E$PTRM
                E$BKEY        E$BUFD
                E$UIUS (bounce package only)
                E$FIUS (bounce package only)
                E$OLDP (old partition only)
```

SAVE$$ -- Save P3ØØ Memory Image as a File

```
   CALL SAVE$$ (RVEC, NAME, NAMLEN, CODE)          (*1522)

   CALL SAVE$ (RVEC, NAME, NAMLEN, LOC(CODE))      (1422)

   CALL SAVE (RVEC, NAME)                          (Ø1Ø2)
```

SEM$DR -- Drain Semaphore

```
   CALL SEM$DR (SEMNUM, CODE)                      (* --)

                ****** CODES  ******
                       E$BPAR
```

SEM$NF -- Notify User Semaphore

```
   CALL SEM$NF (SEMNUM, CODE)                      (* --)

                ****** CODES  ******
                       E$BPAR
                       E$SEMO
```

SEM$TN -- Setup Semaphore for Timed Notifies

```
CALL SEM$TN (SEMNUM, INT32, INT32, CODE)        (* —)

                ****** CODES  ******
                       E$BPAR
                       E$NTIM
```

SEM$TS -- Obtain Current Semaphore Value

```
CALL SEM$TS (SEMNUM, CODE (INT FCN))            (* —)

                ****** CODES  ******
                       E$BPAR
```

SEM$WT -- Wait on User Semaphore

```
CALL SEM$WT (SEMNUM, CODE)                      (* —)

                ****** CODES  ******
                       E$BPAR
```

SGDR$$ -- Position and Read Segment Directory Entries

```
CALL SGDR$$ (KEY, FUNIT, ENTRYA, ENTRYB,
             CODE)                             (*1512)

CALL SEGDR$ (KEY, UNIT, ENTRYA, ENTRYB,
             LOC(CODE))                        (1412)
```

****** KEY ******

K$SPOS = :1	POSITION TO ENTRY NUM IN SEGDIR
K$GOND = :2	POSITION TO END OF SEGDIR
K$GPOS = :3	RETURN CURRENT ENTRY NUMBER
K$MSIZ = :4	MAKE SEGDIR GIVEN NR OF ENTRIES
K$MVNT = :5	MOVE FILE ENTRY TO DIFFERENT
K$FULL = :6	RETURN NEXT NONEMPTY ENTRY
k$FREE = :7	RETURN NEXT FREE ENTRY POSITION

SLEEP$ -- Suspend Execution

```
CALL SLEEP$ (INT32)                             (* --)

INT32 = number of milliseconds to delay
```

SPAS$$ -- Set UFD Passwords

```
CALL SPAS$$ (OPASS, NPASS, CODE)                (*1513)

CALL SPASS$ (OPASS, NPASS, LOC(CODE))           (1413)

                ****** CODES   ******
                E$EOF          E$NOF
                E$UNOP         E$DKFL
                E$NRIT         E$FNTS
                E$EXST         E$DISK
                E$PTRM         E$BKEY
                E$BUNT
```

SRCH$$ -- Open or Close a File

```
CALL SRCH$$ (KEY, NAME, NAMLEN, UNIT, TYPE,
             CODE)                              (*1511)

CALL SEARC$ (KEY, NAME, NAMLEN, UNIT, TYPE,
             LOC(CODE))                         (1411)

CALL SEARCH (KEY, NAME, UNIT, ALTRTN)           (Ø1Ø1)

     KEY = ACTION + REF+ NEWFIL
```

	****** ACTION ******
K$READ = :1	OPEN FOR READ
K$WRIT = :2	OPEN FOR WRITE
K$RDWR = :3	OPEN FOR READING AND WRITING
K$CLOS = :4	CLOSE FILE UNIT
K$DELE = :5	DELETE FILE
K$EXST = :6	CHECK FILE'S EXISTENCE
	****** REF ******
K$IUFD = :Ø	FILE ENTRY IN UFD
K$ISEG = :1ØØ	FILE ENTRY IN SEGMENT DIRECTORY
K$CACC = :1ØØØ	CHANGE ACCESS
	****** NEWFIL ******
K$NSAM = :Ø	NEW SAM FILE
K$NDAM = :2ØØØ	NEW DAM FILE
K$NSGS = :4ØØØ	NEW SAM SEGMENT DIRECTORY
K$NSGD = :6ØØØ	NEW DAM SEGMENT DIRECTORY
K$CURR = :177777	CURRENTLY ATTACHED UFD

```
       ****** CODES  ******
  E$NATT       E$DKFL       E$NRIT
  E$FDEL       E$NTUD       E$NTSD
  E$FNTF       E$BNAM       E$DNTE
  E$DISK       E$BDAM       E$PTRM
  E$BKEY       E$BUNT       E$BSUN
  E$SUNO       E$BUFD
  E$UNOP (for K$CACC KEY only)
  E$UIUS (not for E$EXST KEY)
  E$FDFL (old partition only)
  E$FIUS (not for E$EXST KEY)
```

T$AMLC -- AMLC Reveive/Transmit

```
   CALL T$AMLC (LINE, LOC(BUFF), NW, INST,
               STATV)                          (*Ø513)
```

T$CMPC -- Move Card of Info to User's Space

```
   CALL T$CMPC (UNIT, LOC(BUFF), NW, INST,
               STATV)                          (*Ø512)
```

T$LMPC -- Print Data

```
   CALL T$LMPC (UNIT, LOC(BUFF), NW, INST,
               STATV)                          (*Ø511)
```

T$PMPC -- Punch Data

```
   CALL T$PMPC (UNIT, LOC(BUFF), NW, INST,
               STATV)                          (*Ø515)
```

T$MT -- Move Raw Data from Magtape

```
   CALL T$MT (UNIT, LOC(BUFF), NW, INST, STATV)  (*Ø51Ø)
```

T$VG -- Print Data on Versatec Printer

```
   CALL T$VG (UNIT, LOC(BUFF), NW, INST, STATV)  (*Ø514)
```

T$SLC -- Performs I/O Over SMLC Lines

```
   CALL T$SLC (KEY, LINE, LOC(BUFF), NW)         (1ØØ1)
```

TIMDAT --

```
   CALL TIMDAT (BUFF, BUFLEN)                    (*Ø5Ø2)
```

```
   TIMDAT returns information in BUFF as follows:
     BUFF(1)   Two ASCII characters representing month
         (2)   Two ASCII characters representing day.
         (3)   Two ASCII characters representing year.
         (4)   Integer time in minutes since midnight.
         (5)   Integer time in seconds.
         (6)   Integer time in ticks.
         (7)   Integer CPU time used in seconds.
         (8)   Integer CPU time used in ticks.
         (9)   Integer disk I/O time used in seconds.
         (1Ø)  Integer disk I/O time used in ticks.
         (11)  Integer number of ticks per second.
         (12)  user number.
         (13)  6-character login name, left justified.
         (14)
         (15)
```

TNOU -- Output CHARCNT Chars with CR and LF

```
   CALL TNOU (MSG, CHARCNT)                      (*Ø7Ø2)
```

TNOUA -- Output CHARCNT Characters

```
   CALL TNOUA (MSG, CHARCNT)                     (*Ø7Ø3)
```

TRNMIT -- Transmit Message to a Remote Machine

```
   CALL TRNMIT (LUN, LOC(BUFF), CNT, STATV)      (Ø4Ø5)
```

UNLINK -- Disconnect All Logical Unit Numbers

```
   CALL UNLINK                                   (Ø411)
```

WREC -- Write Record from Memory to Disk

```
CALL WREC (LOC(BUFF), BUFLEN, NW, RA, PDEV,
          ALTRTN)                               (Ø5Ø1)

CALL WRECL (LOC(BUFF), BUFLEN, NW, RA32, PDEV,
           ALTRTN)                              (Ø517
```

WTLIN$ --

```
CALL WTLIN$ (UNIT, LINE, NW, CODE)              (*1526)

CALL WTLIN (UNIT, LINE, NW, ALTRTN)             (Ø2Ø3)
```

SVC NUMBERS

* => PCLable

* --	SEM$DR
* --	SEM$NF
* --	SEM$TN
* --	SEM$TS
* --	SEM$WT
* --	SLEEP$
Ø1ØØ	ATTACH
Ø1Ø1	SEARCH
Ø1Ø2	SAVE
Ø1Ø3	RESTOR
Ø1Ø4	RESUME
*Ø1Ø5	EXIT
Ø1Ø6	ERRRTN
Ø11Ø	GETERR
Ø111	PRERR
Ø112	GINFO
Ø113	CNAME
Ø114	ERRSET
*Ø115	FORCEW
Ø2Ø2	RDLIN
Ø2Ø3	WTLIN
Ø3ØØ	PWRFIL
Ø4ØØ	FAMSVC
Ø4Ø1	CONECT
Ø4Ø2	GETCON
Ø4Ø3	RJCON
Ø4Ø4	RECEIV
Ø4Ø5	TRNMIT
Ø4Ø6	NETWAT
Ø4Ø7	NTSTAT
Ø41Ø	DISCON
Ø411	UNLINK
Ø412	NETLNK
Ø5ØØ	RREC
Ø5Ø1	WREC
*Ø5Ø2	TIMDAT
*Ø5Ø5	RECYCL
Ø5Ø6	D$INIT
*Ø5Ø7	BREAK$
*Ø51Ø	T$MT
*Ø511	T$LMPC
*Ø512	T$CMPC
*Ø513	T$AMLC
*Ø514	T$VG
*Ø515	T$PMPC
Ø516	RRECL
Ø517	WRECL

*Ø6ØØ	COMANL	Ø6)
*Ø6Ø1	C1IN	
Ø6Ø2	CMREAD	
Ø6Ø3	COMINP	6)
*Ø6Ø4	CNIN$	
*Ø7Ø2	TNOU	Ø)
*Ø7Ø3	TNOUA	
*Ø7Ø5	DUPLX$	
1ØØ1	T$SLC	
14ØØ	ATTAC$	
14Ø1	CREAT$	
*14Ø2	ERRPR$	
14Ø4	GPASS$	
14Ø6	PRWFL$	
14Ø7	RDENT$	
141Ø	SATTR$	
1411	SEARC$	
1412	SEGDR$	
1413	SPASS$	
1415	CNAME$	
1416	COMIN$	
1417	RDTKN$	
142Ø	RESTO$	
1421	RESUM$	
1422	SAVE$	
*15ØØ	ATCH$$	
*15Ø1	CREA$$	
*15Ø4	GPAS$$	
*15Ø6	PRWF$$	
*15Ø7	RDEN$$	
*151Ø	SATR$$	
*1511	SRCH$$	
*1512	SGDR$$	
*1513	SPAS$$	
*1515	CNAM$$	
*1516	COMI$$	
*1517	RDTK$$	
*152Ø	REST$$	
*1521	RESU$$	
*1522	SAVE$$	
*1523	COMO$$	
*1524	ERKL$$	
*1525	RDLIN$	
*1526	WTLIN$	

ERROR MESSAGES AND CODES (SYSCOM>ERRD.F)

E$EOF= 1	END OF FILE	PE
E$BOF= 2	BEGINNING OF FILE	PG
E$UNOP= 3	UNIT NOT OPEN	PD,SD
E$UIUS= 4	UNIT IN USE	SI
E$FIUS= 5	FILE IN USE	SI
E$BPAR= 6	BAD PARAMETER	SA
E$NATT= 7	NO UFD ATTACHED	SL,AL
E$FDFL= 8	UFD FULL	SK
E$DKFL= 9	DISK FULL	DJ
E$NRIT=1Ø	NO RIGHT	SX
E$FDEL=11	FILE OPEN ON DELETE	SD
E$NTUD=12	NOT A UFD	AR
E$NTSD=13	NOT A SEGDIR	--
E$DIRE=14	IS A DIRECTORY	--
E$FNTF=15	(FILE) NOT FOUND	SH,AH
E$FNTS=16	(FILE) NOT FOUND IN SEGDIR	SQ
E$BNAM=17	ILLEGAL NAME	CA
E$EXST=18	ALREADY EXISTS	CZ
E$DNTE=19	DIRECTORY NOT EMPTY	--
E$SHUT=2Ø	BAD SHUTDN (FAM ONLY)	BS
E$DISK=21	DISK I/O ERROR	WB
E$BDAM=22	BAD DAM FILE (FAM ONLY)	SS
E$PTRM=23	PTR MISMATCH (FAM ONLY)	PC,DC,AC
E$BPAS=24	BAD PASSWORD (FAM ONLY)	AN
E$BCOD=25	BAD CODE IN ERRVEC	--
E$BTRN=26	BAD TRUNCATE OF SEGDIR	--
E$OLDP=27	OLD PARTITION	--
E$BKEY=28	BAD KEY	--
E$BUNT=29	BAD UNIT NUMBER	--
E$BSUN=3Ø	BAD SEGDIR UNIT	SA
E$SUNO=31	SEGDIR UNIT NOT OPEN	--
E$NMLG=32	NAME TOO LONG	--
E$SDER=33	SEGDIR ERROR	SQ
E$BUFD=34	BAD UFD	--
E$BFTS=35	BUFFER TOO SMALL	--
E$FITB=36	FILE TOO BIG	--
E$NULL=37	(NULL MESSAGE)	--
E$IREM=38	ILL REMOTE REF	--
E$DVIU=39	DEVICE IN USE	--
E$RLDN=4Ø	REMOTE LINE DOWN	--
E$FUIU=41	ALL REMOTE UNITS IN USE	--
E$DNS=42	DEVICE NOT STARTED	--
E$TMUL=43	TOO MANY UFD LEVELS	--
E$FBST=44	FAM - BAD STARTUP	--
E$BSGN=45	BAD SEGMENT NUMBER	--
E$FIFC=46	INVALID FAM FUNCTION CODE	--
E$TMRU=47	MAX REMOTE USERS EXCEEDED	--
E$NASS=48	DEVICE NOT ASSIGNED	--
E$BFSV=49	BAD FAM SVC	--
E$SEMO=5Ø	SEM OVERFLOW	--
E$NTIM=51	NO TIMER	--
E$FABT=52	FAM ABORT	--
E$FONC=53	FAM OP NOT COMPLETE	--

Ø6)

6)

Ø)

10 APPENDICES

ASCII CHARACTER SET

F => Valid file name char
R => Rsrved cmd line char
^ => ctrl key depressed

8-Bit Octal Code	Char		Code in Left Byte	8-Bit Octal Code	Char	Code in Left Byte
200	NUL		1000	240	Sp	1200
201	SOH	^A	1004	241	!	1204 R
202	STX	^B	1010	242	"	1210 R
203	ETX	^C	1014	243	#	1214 F
204	EOT	^D	1020	244	$	1220 F
205	ENQ	^E	1024	245	%	1224 R
206	ACK	^F	1030	246	&	1230 FR
207	BEL	^G	1034	247	'	1234 R
210	BS	^H	1040	250	(	1240 R
211	HT	^I	1044	251	)	1244 R
212	NL	^J	1050	252	*	1250 F
213	VT	^K	1054	253	+	1254
214	FF	^L	1060	254	,	1260 R
215	CR	^M	1064	255	-	1264 FR
216	RRS	^N	1070	256	.	1270 F
217	BRS	^O	1074	257	/	1274 F
220	RCP	^P	1100	260	0	1300 F
221	RHT	^Q	1104	261	1	1304 F
222	HLF	^R	1110	262	2	1310 F
223	RVT	^S	1114	263	3	1314 F
224	HLR	^T	1120	264	4	1320 F
225	NAK	^U	1124	265	5	1324 F
226	SYN	^V	1130	266	6	1330 F
227	ETB	^W	1134	267	7	1334 F
230	CAN	^X	1140	270	8	1340 F
231	EM	^Y	1144	271	9	1344 F
232	SUB	^Z	1150	272	:	1350 R
233	ESC	^UP-K	1154	273	;	1354 R
234	FS	^UP-L	1160	274	<	1360
235	GS	^UP-M	1164	275	=	1364 R
236	RS	^UP-N	1170	276	>	1370
237	US	^UP-O	1174	277	?	1374

8-Bit Octal Code	Char	Code in Left Byte	8-Bit Octal Code	Char	Code in Left Byte
300		1400 R	340		1600 R
301	A	1404 F	341	a	1604
302	B	1410 F	342	b	1610
303	C	1414 F	343	c	1614
304	D	1420 F	344	d	1620
305	E	1424 F	345	e	1624
306	F	1430 F	346	f	1630
307	G	1434 F	347	g	1634
310	H	1440 F	350	h	1640
311	I	1444 F	351	i	1644
312	J	1450 F	352	j	1650
313	K	1454 F	353	k	1654
314	L	1460 F	354	l	1660
315	M	1464 F	355	m	1664
316	N	1470 F	356	n	1670
317	O	1474 F	357	o	1674
320	P	1500 F	360	p	1700
321	Q	1504 F	361	q	1704
322	R	1510 F	362	r	1710
323	S	1514 F	363	s	1714
324	T	1520 F	364	t	1720
325	U	1524 F	365	u	1724
326	V	1530 F	366	v	1730
327	W	1534 F	367	w	1734
330	X	1540 F	370	x	1740
331	Y	1544 F	371	y	1744
332	Z	1550 F	372	z	1750
333	[	1554 R	373	{	1754 R
334	\	1560 R	374	|	1760
335	]	1564 R	375	}	1764 R
336	^	1570 R	376	~	1770 R
337	_	1574 F	377	DEL	1774

CONVERSION TABLES

OCTAL DECIMAL CONVERSION TABLE

0	0	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14	15
20	16	17	18	19	20	21	22	23
30	24	25	26	27	28	29	30	31
40	32	33	34	35	36	37	38	39
50	40	41	42	43	44	45	46	47
60	48	49	50	51	52	53	54	55
70	56	57	58	59	60	61	62	63
100	64	65	66	67	68	69	70	71
110	72	73	74	75	76	77	78	79
120	80	81	82	83	84	85	86	87
130	88	89	90	91	92	93	94	95
140	96	97	98	99	100	101	102	103
150	104	105	106	107	108	109	110	111
160	112	113	114	115	116	117	118	119
170	120	121	122	123	124	125	126	127
200	128	129	130	131	132	133	134	135
210	136	137	138	139	140	141	142	143
220	144	145	146	147	148	149	150	151
230	152	153	154	155	156	157	158	159
240	160	161	162	163	164	165	166	167
250	168	169	170	171	172	173	174	175
260	176	177	178	179	180	181	182	183
270	184	185	186	187	188	189	190	191
300	192	193	194	195	196	197	198	199
310	200	201	202	203	204	205	206	207
320	208	209	210	211	212	213	214	215
330	216	217	218	219	220	221	222	223
340	224	225	226	227	228	229	230	231
350	232	233	234	235	236	237	238	239
360	240	241	242	243	244	245	246	247
370	248	249	250	251	252	253	254	255
400	256	257	258	259	260	261	262	263
410	264	265	266	267	268	269	270	271
420	272	273	274	275	276	277	278	279
430	280	281	282	283	284	285	286	287
440	288	289	290	291	292	293	294	295
450	296	297	298	299	300	301	302	303
460	304	305	306	307	308	309	310	311
470	312	313	314	315	316	317	318	319
500	320	321	322	323	324	325	326	327
510	328	329	330	331	332	333	334	335
520	336	337	338	339	340	341	342	343
530	344	345	346	347	348	349	350	351
540	352	353	354	355	356	357	358	359
550	360	361	362	363	364	365	366	367
560	368	369	370	371	372	373	374	375
570	376	377	378	379	380	381	382	383
600	384	385	386	387	388	389	390	391
610	392	393	394	395	396	397	398	399
620	400	401	402	403	404	405	406	407
630	408	409	410	411	412	413	414	415
640	416	417	418	419	420	421	422	423
650	424	425	426	427	428	429	430	431
660	432	433	434	435	436	437	438	439
670	440	441	442	443	444	445	446	447
700	448	449	450	451	452	453	454	455
710	456	457	458	459	460	461	462	463
720	464	465	466	467	468	469	470	471
730	472	473	474	475	476	477	478	479
740	480	481	482	483	484	485	486	487
750	488	489	490	491	492	493	494	495
760	496	497	498	499	500	501	502	503
770	504	505	506	507	508	509	510	511
1000	512	513	514	515	516	517	518	519
1010	520	521	522	523	524	525	526	527
1020	528	529	530	531	532	533	534	535
1030	536	537	538	539	540	541	542	543
1040	544	545	546	547	548	549	550	551
1050	552	553	554	555	556	557	558	559
1060	560	561	562	563	564	565	566	567
1070	568	569	570	571	572	573	574	575
1100	576	577	578	579	580	581	582	583
1110	584	585	586	587	588	589	590	591
1120	592	593	594	595	596	597	598	599
1130	600	601	602	603	604	605	606	607
1140	608	609	610	611	612	613	614	615
1150	616	617	618	619	620	621	622	623
1160	624	625	626	627	628	629	630	631
1170	632	633	634	635	636	637	638	639
1200	640	641	642	643	644	645	646	647
1210	648	649	650	651	652	653	654	655
1220	656	657	658	659	660	661	662	663
1230	664	665	666	667	668	669	670	671
1240	672	673	674	675	676	677	678	679
1250	680	681	682	683	684	685	686	687
1260	688	689	690	691	692	693	694	695
1270	696	697	698	699	700	701	702	703
1300	704	705	706	707	708	709	710	711
1310	712	713	714	715	716	717	718	719
1320	720	721	722	723	724	725	726	727
1330	728	729	730	731	732	733	734	735
1340	736	737	738	739	740	741	742	743
1350	744	745	746	747	748	749	750	751
1360	752	753	754	755	756	757	758	759
1370	760	761	762	763	764	765	766	767
1400	768	769	770	771	772	773	774	775
1410	776	777	778	779	780	781	782	783
1420	784	785	786	787	788	789	790	791
1430	792	793	794	795	796	797	798	799
1440	800	801	802	803	804	805	806	807

0	0	1	2	3	4	5	6	7
1450	808	809	810	811	812	813	814	815
1460	816	817	818	819	820	821	822	823
1470	824	825	826	827	828	829	830	831
1500	832	833	834	835	836	837	838	839
1510	840	841	842	843	844	845	846	847
1520	848	849	850	851	852	853	854	855
1530	856	857	858	859	860	861	862	863
1540	864	865	866	867	868	869	870	871
1550	872	873	874	875	876	877	878	879
1560	880	881	882	883	884	885	886	887
1570	888	889	890	891	892	893	894	895
1600	896	897	898	899	900	901	902	903
1610	904	905	906	907	908	909	910	911
1620	912	913	914	915	916	917	918	919
1630	920	921	922	923	924	925	926	927
1640	928	929	930	931	932	933	934	935
1650	936	937	938	939	940	941	942	943
1660	944	945	946	947	948	949	950	951
1670	952	953	954	955	956	957	958	959
1700	960	961	962	963	964	965	966	967
1710	968	969	970	971	972	973	974	975
1720	976	977	978	979	980	981	982	983
1730	984	985	986	987	988	989	990	991
1740	992	993	994	995	996	997	998	999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

POWERS OF TWO

POSITIVE POWERS OF TWO

n	2
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	13107 2
18	26214 4
19	52428 8
20	10485 76
21	20971 52
22	41943 04
23	83886 08
24	16777 216
25	33554 432
26	67108 864
27	13421 7728
28	26843 5456
29	53687 0912
30	10737 41824
31	21474 83648
32	42949 67296
33	85899 34592
34	17179 86918 4
35	34359 73836 8
36	68719 47673 6
37	13743 89534 72
38	27487 79069 44
39	54975 58138 88
40	10995 11627 776
41	21990 23255 552
42	43980 46511 104
43	87960 93022 208
44	17592 18604 4416
45	35184 37208 8832
46	70368 74417 7664
47	14073 74883 55328
48	28147 49767 10656

NEGATIVE POWERS OF TWO

n	2					
0	1.0					
1	0.5					
2	0.25					
3	0.125					
4	0.0625					
5	0.03125					
6	0.01562	5				
7	0.00781	25				
8	0.00390	625				
9	0.00195	3125				
10	0.00097	65625				
11	0.00048	82812	5			
12	0.00024	41406	25			
13	0.00012	20703	125			
14	0.00006	10351	5625			
15	0.00003	05175	78125			
16	0.00001	52587	89062	5		
17	0.00000	76293	94531	25		
18	0.00000	38146	97265	625		
19	0.00000	19073	48632	8125		
20	0.00000	09536	74316	40625		
21	0.00000	04768	37158	20312	5	
22	0.00000	02384	18579	10156	25	
23	0.00000	01192	09289	55078	125	
24	0.00000	00596	04644	77539	0625	
25	0.00000	00298	02322	38769	53125	
26	0.00000	00149	01161	19384	76562	5
27	0.00000	00074	50580	59692	38281	25
28	0.00000	00037	25290	29846	19140	625
29	0.00000	00018	62645	14923	09570	3125
30	0.00000	00009	31322	57461	54785	15625
31	0.00000	00004	65661	28730	77392	57812
32	0.00000	00002	32830	64356	28696	28906
33	0.00000	00001	16415	32182	69348	14453
34	0.00000	00000	58207	66091	34674	07226
35	0.00000	00000	29103	83045	67337	03613
36	0.00000	00000	14551	91522	83668	51806
37	0.00000	00000	07275	95761	41834	25903
38	0.00000	00000	03637	97880	70917	12951
39	0.00000	00000	01818	98940	35458	56475
40	0.00000	00000	00909	49470	17629	28237
41	0.00000	00000	00454	74735	08864	64118
42	0.00000	00000	00227	37367	54432	32059
43	0.00000	00000	00113	68683	77216	16029
44	0.00000	00000	00056	84341	88608	08014

11 GLOSSARY

ALTRTN
Alternate return.

BUFF
Buffer (usually INTEGER*2 array).

AP
argument pointer.

code
a value returned by a routine indicating either the success of or the reason for failure to accomplish the requested action.

command
a program called from command level. (See internal command and external command, below.)

command level
the process state in which input lines are interpreted by Primos as commands. A process is at command level when a user logs in and when a command either completes, encounters an error, or stops after a quit signal is issued.

crash
an unplanned interruption of system availability caused by problems in hardware and/or software.

CRASH ADDR
displacement in hardware register save area (pointed to by RSAVPTR -- R37).

CRS
Current register set.

DAM file
a direct access method file.

directory
a catalog of files and other subordinate directories. See MFD, UFD, and segment directory.

DSW
diagnostic status word.

DTAR
descriptor table address register.

DVNO
physical device number.

ECB
Entry Control Block.

external command
a command that executes in user address space.

FADDRH
fault address, high.

FADDRL
fault address, low.

fault
a hardware or softwre condition that causes system failure.

FCODE
fault code.

file
a sam or dam file.

filename
a name given to an item in a directory.

FUNIT
A file system unit number Ø-'21.

internal command
a command that executes in the address space occupied by the Primos operating system.

interrupt
A signal received from a device in the external world (including clocks) indicating that the device either needs to be serviced or has completed an operation.

LB
linkage base register.

login name
A 6-character name by which each logged-in user is known to PRIMOS.

LSB
Least Significant Bit.

MFD
master file directory. An MFD contains information about each UFD on the disk.

MSB
Most Significant Bit.

NODE
the name of a system connected to PRIMENET.

NW

Number of words.

non-owner, NPASS

Non-owner password.

owner, OPASS

Owner password.

page

a 1Ø24 16-bit word block of data within a segment.

page control

the routines that manage the transfer of pages between secondary storage and main memory frames.

paged address space

nonvisible memory; allocation of physical memory.

password

a character string supplied by the user that controls his access to various files and directories. See owner and non-owner.

PB

procedure base register.

PBH

procedure base register, high side.

PBL

procedure base register, low side.

PCB

see process control block.

PDEV

physical device number.

physical volume

a disk pack.

PMNT

page map entry.

pointer

an address value either 16-bits or 32-bits in length.

procedure control block

64-word block describing current state of a process.

process

an address space and an execution point. Each logged-in user has his own process.

register file

128 32-bit registers partitioned into 4 32-register blocks. The first block is reserved for microcode use, the second block is used for DMA channels, the third and fourth blocks are process register sets.

RFIL, RFILE

Register file address.

ring

a level of privilege at which programs execute. Supervisor programs run in ring Ø; most user programs run in ring 3.

SAM file

a sequential access method file.

SB

stack base pointer.

SDW

Segment Descriptor Word.

segment directory

a directory that contains nothing but pointers to the first record of each file cataloged in it.

semaphore

a special purpose integer variable allocated in the universe in which the processes are embedded, to perform explicit mutual synchronization of parallel sequential processes.

segmented address space

visible memory; allocation of user's data/procedures.

stack

a pushdown list where active procedures maintain private regions used for temporary variables.

STATV

A three word vector used by the T$xxxx routines. STATV(1) set to Ø when operation completes, STATV(2) = device status, STATV(3) = number words transferred.

TRAP

An asynchronous interruption of sequential microcode execution.

treename, tree

a character string that specified a file by its position in the file system hierarchy. Valid treenames are of the form:

[<ldev>]ufdname[password][>ufdname ...]>filename

Use of < and > in a treename are literal and must be typed as shown. If the treename includes embedded blanks and is entered at PRIMOS command level, it should be surrounded by apostrophes.

UFD
User File Directory. An UFD contains information about the location and content of each file cataloged in it.

UNIT
See FUNIT.

wired page
a page that remains in main memory at all times.

word
a unit of information that is 16 bits in length.

XB
temporary base register.

12 INDEX